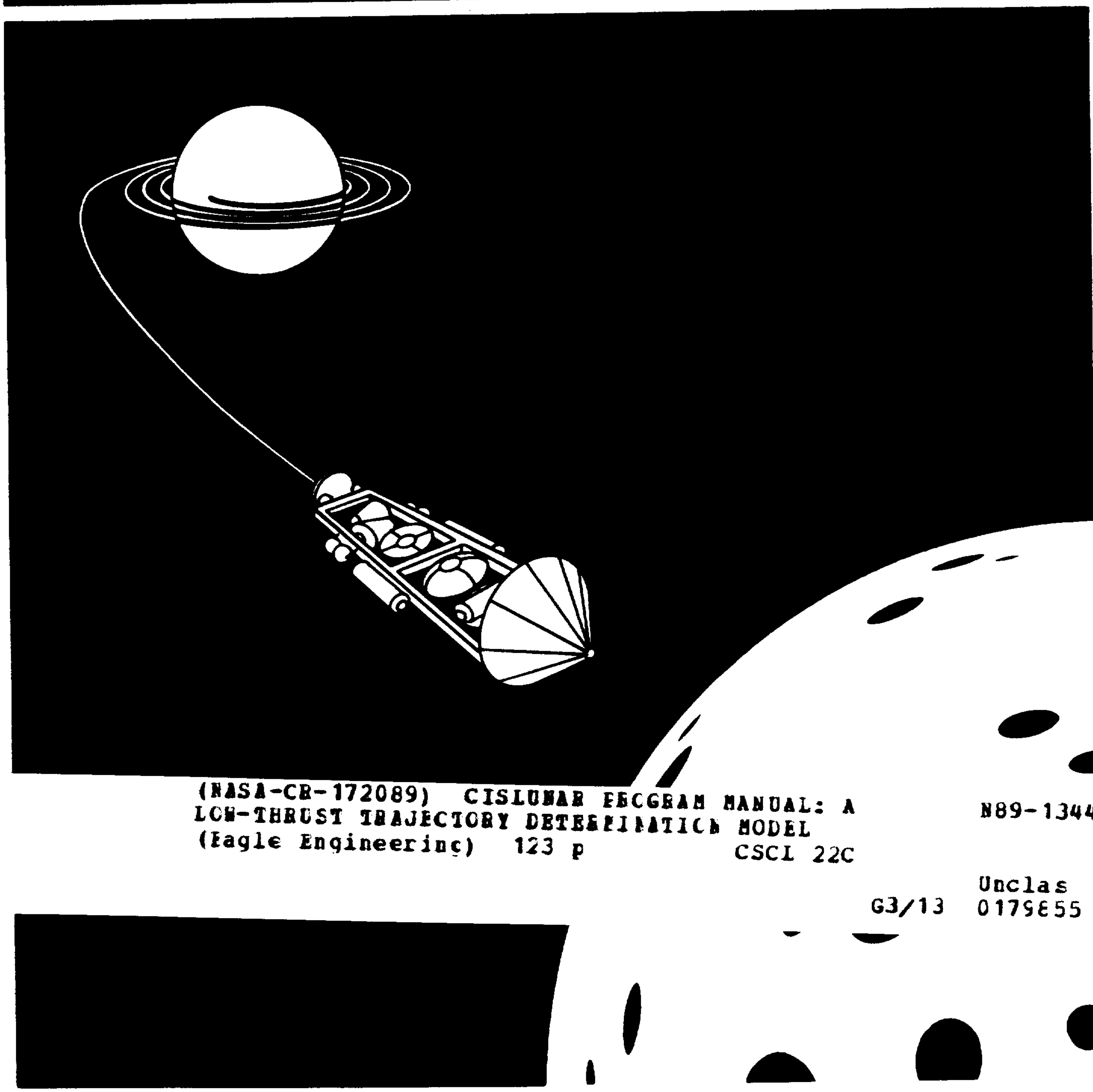
LBSS CISLUNAR Program Manual



(NASA-CR-172089) CISLUNAR PROGRAM MANUAL: A LOW-THRUST TRAJECTORY DETERMINATION MODEL (Eagle Engineering) 123 p CSCL 22C

N89-13444

Unclas
G3/13 0179855

NASA Contract Number NAS 9-17878
EEI Report Number 88-209
September 30,1988

EAGLE

CISLUNAR
Program Manual

A Low-Thrust Trajectory
Determination Model

Prepared for the
National Aeronautics and Space Administration
Johnson Space Center
Advanced Programs Office
as part of the
Advanced Space Transportation Support Contract (ASTS)
and the
Lunar Base Systems Study (LBSS)

Contract Number: NAS 9-17878

by
Eagle Engineering, Incorporated
Report Number: 88-209

September 30, 1988

FOREWORD

This report was prepared between June 1988 and September 1988 by Eagle Engineering, Inc. for the Advanced Programs Office of Johnson Space Center, a field center of the National Aeronautics and Space Administration. The objective is to provide a program which can be used to analyze the performance of a spacecraft making a low-thrust flight between the Earth and the Moon.

Dr. J.W. Alred was the NASA technical monitor for the Advanced Space Transportation Study contract of which this task was a part. Mr. Andy Petro was the NASA task monitor for this particular task. Mr. W.R. Stump was the Eagle project manager. Mr. C.C. Varner was the Eagle task manager for this task. This program was originally written and documented in BASIC by Mr. D.J. Korsmeyer of the Large Scale Programs Institute at the University of Texas. The conversion to FORTRAN was completed by Mr. M. D'Onofrio of Eagle, and FORTRAN documentation was prepared by Mr. C.C Varner.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 USING CISLUNAR

CISLUNAR is a stand alone program designed to generate the trajectory of a low-thrust spacecraft travelling in Earth-Moon space. The program allows the creation of functional trajectories dependent upon the supplied spacecraft characteristics. The trajectory generation is a user interactive process. The original intent was for the program user to modify the necessary control values until a satisfactory trajectory has been created.

The program is started by simply typing *CISLUNAR*. The information that appears on the screen indicates that *CISLUNAR* has started, and shows the spacecraft's default characteristics. These characteristics can be modified by the user at the beginning of each run. The program prompts the user for the direction of the trajectory generation by asking whether the initial orbit is about the Earth or the Moon. This sets the direction flags for the rest of the program. The next question is whether new parametric velocity curves based on the spacecraft's characteristics should be created. If this question is answered "yes", the program generates these curves before continuing. The next three questions concern the initial altitude, velocity, and orbital position of the spacecraft. The altitude must be input for the program to continue. The velocity will default to the circular velocity at the input altitude. The final question asks if the controls for the spacecraft generation need to be modified. If this question is not answered or the answer is "no", the program uses the default values for the controls.

Four controls are specified, Jac1, Jac2, Jac3, and Range. These four values govern the thrusting of the spacecraft during the final escape and translunar portions of the trajectory. Jac1 indicates the spacecraft is nearing the end of its spiral escape from the initial orbit; the engines shut down, and thrusting ceases unless the spacecraft is in the proper quadrant for transfer injection. Jac2 is the control that determines whether the spacecraft can achieve a cislunar trajectory. Ideally the Jacobian Constant at Jac2 has a value of $3 \pm .2$ <km/s>. After reaching Jac2, the spacecraft thrusts continuously. Jac3 is the final constraint on the amount of energy that is to be supplied to the spacecraft during transfer. Following Jac3, the spacecraft does not thrust. Range is the control that determines the distance from the initial planet that the capture guidance to the target planet is begun. This is the point at which reverse thrusting begins.

Markers for these four controls show up on the trajectory as each of them is passed. For Jac1-Jac3, a small circle will indicate that this control has been reached. The passage of Range is indicated by a small vertical line. The visual representation of the controls is helpful to understand and plan a modification of the controls. The markers do not appear in FORTRAN versions of the program.

While the trajectory is being generated the program can be paused, restarted, or simply stopped at any time.

2.0 PROGRAM CONSIDERATIONS

The trajectory determination methods for impulsive and low-thrust spacecraft differ considerably. Electric propulsion systems need to thrust continuously for long periods of time in order to achieve a significant velocity change. Chemical, or impulsive propulsion, can create a near instantaneous change in the velocity. Where an impulsive thrusting spacecraft could use two short powerful thrusts to transfer between LEO (Low Earth Orbit) and a higher orbit, a low-thrust orbital transfer starting in LEO would be accomplished as a very slow outward spiral to the desired altitude. The fractional increase of the orbital radius per revolution is very small for low-thrust spacecraft. This complicates the calculation of trajectories for low-thrust vehicles.

The characteristics of the low-thrust spacecraft will also play an important role in determining the type of trajectory that can be developed. The vehicle's propulsion system and associated power system will have a considerable impact on the type of trajectories available.

The power and propulsion system for a low-thrust spacecraft are intimately coupled. The thruster system efficiency is the fraction of electrical power that is converted to exhaust kinetic energy. This yields,

$$\eta = \frac{\dot{m}(I_{sp}g)^2}{2\ P_o} \qquad \{2.1\}$$

where η is the thruster system efficiency, $\dot{m}$ is the mass flow rate of the thrusters, and P_o is the electrical power input to the propulsion system.[1]

Currently, there are many different low-thrust electric propulsion systems under investigation. The ion engine, magnetoplasmadynamic thruster, and arcjet are a few of the leading candidates. All of these engines will require continuous high power to be able to perform competitively against chemical propulsion. Nuclear and solar power systems are the major competitors for high power supplies in space. Solar arrays and solar dynamic power systems have the advantage of using the sun as a heat source, however, they require continual sunlight. For some propulsion systems solar power cannot provide the needed level of power. Nuclear power, on the other hand, has tremendous potential for fulfilling the power needs of electric propulsion systems. The proposed range of power available from nuclear sources ranges from a few kilowatts to megawatts.[2]

This program does not include an aerocapture option. Aerocapture has numerous problems for large nuclear or solar power sources. The general outbound trajectory assumed is a spiral out from LEO and a spiral down into LLO (Low Lunar Orbit). The return trajectory is a spiral up from LLO and then down into LEO.

The guidance scheme employed to determine a trajectory must use only low-thrust to capture the OTV into LEO. The low-thrust OTV is limited in the range of thrusting acceleration available to drive the vehicle to the desired orbit. Another restriction for the trajectories of nuclear-powered OTVs is the proposed nuclear safe orbit (NSO).[3] This would be a designated altitude below which the nuclear powered spacecraft would be prohibited. The spacecraft would be prohibited from descending below this altitude at any point of the trajectory.

In the development of trajectories for low-thrust cislunar OTVs, little attention has been directed at the guidance and control of the spacecraft. The premise that the guidance of the vehicle and the determination of the appropriate trajectory are unrelated is false. Rather guidance and trajectory determination for low thrust vehicles are closely related problems which, by necessity, must be treated with equal importance.[4]

---

1. Hill, P.G., and Peterson, C.R., p. 336.

2. English, Robert E., "Power Generation from Nuclear Reactors in Aerospace Applications, "NRC Symposium on Advanced Compact Reactors, Washington, D.C., November 15-17, 1982.

3. Galecki, Diane L., and Patterson, Michael J., "Nuclear Powered Mars Cargo Transport Mission Utilizing Ion Propulsion," AIAA/SAE/ASME/ASEE 23rd Joint Propulsion Conference, San Diego, CA, 1987, AIAA-87-1903.

4. Battin, R.H., and Miller, J.S., "Trajectories and Guidance Theory for a Continuous Low-Thrust Lunar Reconnaissance Vehicle," 6th Symposium on Ballistic Missile and Aerospace Technology, 1961.

3.0 PROBLEM FORMULATION

A major problem in the design of low-thrust OTVs and their associated trajectories is the lack of an end to end simulation tool for the spacecraft trajectory, from NSO travelling to LLO and the subsequent return. The current concern is how the vehicle will behave at the proposed thrust level and how it will be guided on its trajectory.

To adequately understand the dynamics of motion of the low-thrust spacecraft, the gravitational effects of the Earth and the Moon on the spacecraft must be included for the full duration of the trajectory. The thrusting acceleration for low-thrust OTVs in high Earth orbit is the same magnitude as the perturbing force due to the Moon. To model the Earth-Moon system with the necessary accuracy and achieve computational efficiency, the restricted three-body formulation of the dynamical equations is utilized as the governing equations of motion.

3.1 RESTRICTED THREE-BODY FORMULATION

The problem of three bodies was first formulated in 1772 by Lagrange. Further studies by Poincare, Laplace, Hill and Szebehely have resulted in a detailed treatment of the problem and a general understanding of the interactions between the two primary gravitational fields. Various formulations are available to represent the problem of three bodies. The formulation used in this study was referenced from Kaplan[1] and Moulton.[2]

Many realistic orbital cases may be modelled as restricted three-body situations. An exemplary case is that of a spacecraft moving in the Earth-Moon system. Certain assumptions are made about the nature of the Earth-Moon system that permit a straightforward solution at a slight loss of accuracy. The motion of the Earth-Moon system is assumed to be circular and coplanar about its center of mass (barycenter) and the spacecraft, at point P, has negligible mass. This system is shown in Figure 3.1. The motion of the spacecraft is governed by the relative gravitational attraction of the Earth and the Moon rotating about the barycenter. The spacecraft is assumed to have no impact on the motion of the Earth or the Moon. Thus, the acceleration at P is

$$\mathbf{a}_p = \frac{-\mu_\oplus}{r^3_\oplus}\,\mathbf{r}_\oplus + \frac{\mu_m}{r^3_m}\,\mathbf{r}_m = \nabla \left[\frac{\mu_\oplus}{\mathbf{r}_\oplus} + \frac{\mu_m}{\mathbf{r}_m} \right] \tag{3.1}$$

The absolute acceleration of the spacecraft is obtained in terms of the rotating coordinate system, x,y,z, by relating the acceleration of the spacecraft in the non-inertial (barycenter) rotating system to that in the inertial coordinate system. Hence,

$$\mathbf{a}_p = \mathbf{a}_o + \mathbf{n} \times (\mathbf{n} \times \mathbf{r}) + \ddot{\mathbf{r}}_b + 2\mathbf{n} \times \dot{\mathbf{r}}_b \tag{3.2}$$

where,

$\mathbf{r}$ is the radius vector of the spacecraft,

$\dot{\mathbf{r}}_b$ is the apparent velocity of the spacecraft in the rotating coordinates,

$\ddot{\mathbf{r}}_b$ is the apparent acceleration of the spacecraft in the rotating coordinates,

$\mathbf{a}_p$ is the acceleration of the spacecraft in inertial coordinates,

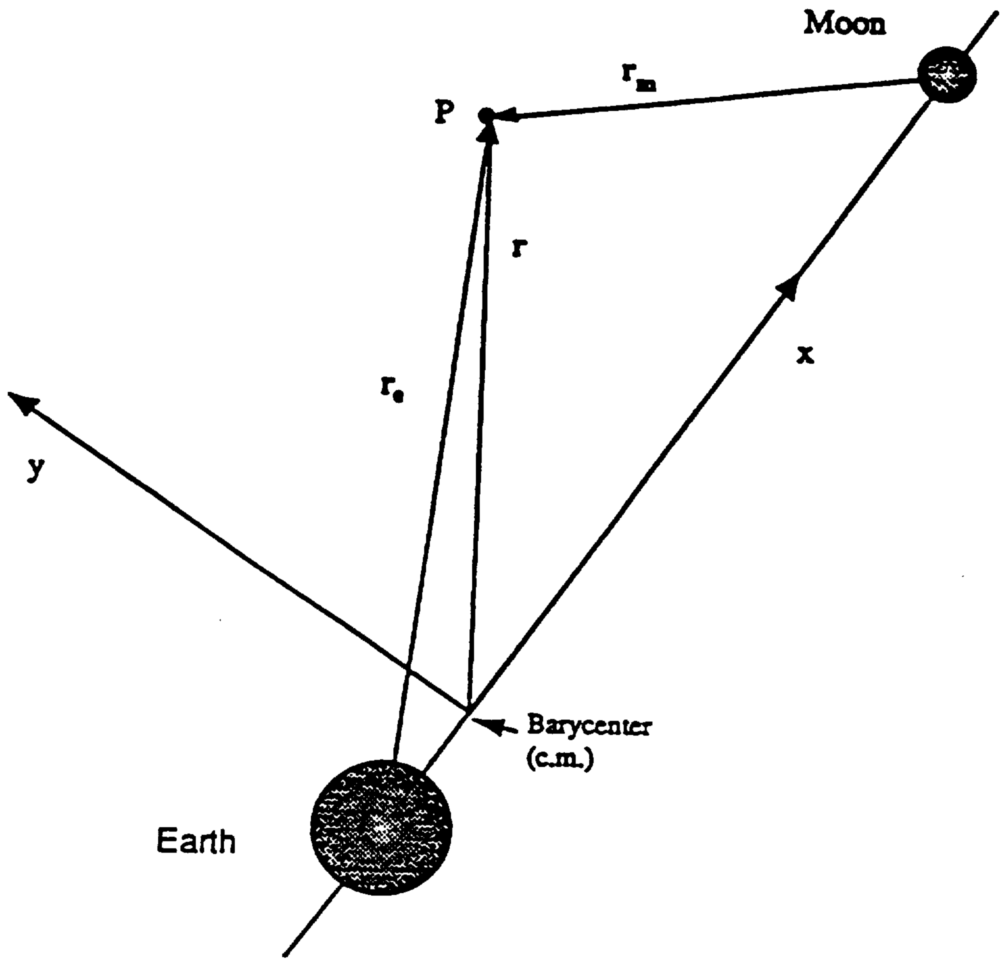


Figure 3.1 Restricted Three-body Problem Nomenclature

$\mathbf{a}_o$ is the acceleration of the origin in the inertial coordinates,

$\mathbf{n}$ is the angular velocity vector of the Earth-Moon system, $n = n\mathrm{i}_z$,

$\mathbf{n} \times (\mathbf{n} \times \mathbf{r})$ is the centrifugal acceleration, and

$2\,\mathbf{n} \times \dot{\mathbf{r}}_b$ is the Coriolis acceleration due to the motion of the spacecraft in x,y,z.

Equating {3.1} and {3.2}, noting that the acceleration of the origin is the same for both equations, and expressing the acceleration in component form, yields the equations of motion for the spacecraft in the rotating coordinate system.

$$\ddot{x} - 2n\dot{y} - n^2x = \frac{\delta}{\delta x}\left[\frac{\mu_e}{r_e} + \frac{\mu_m}{r_m}\right]$$

$$\ddot{y} + 2n\dot{x} - n^2y = \frac{\delta}{\delta y}\left[\frac{\mu_e}{r_e} + \frac{\mu_m}{r_m}\right] \qquad \{3.4\}$$

3.2 JACOBIAN CONSTANT

In this formulation of the equations of motion, the energy of the spacecraft is not conserved. However, the sum of the angular momentum, velocity, and potential energy of the spacecraft is conserved. This can be shown with the Jacobian Integral. Multiplying the first equation of {3.4} by dx/dt, the second by dy/dt, adding, and integrating the result yields this integral.

$$\dot{x}^2 + \dot{y}^2 - n^2(x^2 + y^2) = \frac{2\mu_e}{r_e} + \frac{2\mu_m}{r_m} - C \qquad \{3.5\}$$

where C is known as the Jacobian constant. Mathematician Karl Gustav Jacobi first formulated this integral in 1836. This constant, C, can be determined for any set of initial conditions. Equation 3.5 determines the locus of those points where the spacecraft can travel given the initial conditions. In particular, if the velocity of the vehicle is set equal to zero for a given C, equation

3.5 will describe a curve where the spacecraft's motion is bounded. On this curve the spacecraft with a given C will have zero velocity. Only on the inside of the curve will the square of the spacecraft's velocity be positive, restricting the motion of the vehicle to that side. Figure 3.2 shows a series of zero velocity curves in the Earth-Moon system.

---

1. Kaplan, Marshall H., Page 290-300.

2. Moulton, Forest Ray, An Introduction to Celestial Mechanics. (Dover Publications, Inc., New York: 1914, 2nd Revised Edition) pages 277-287.

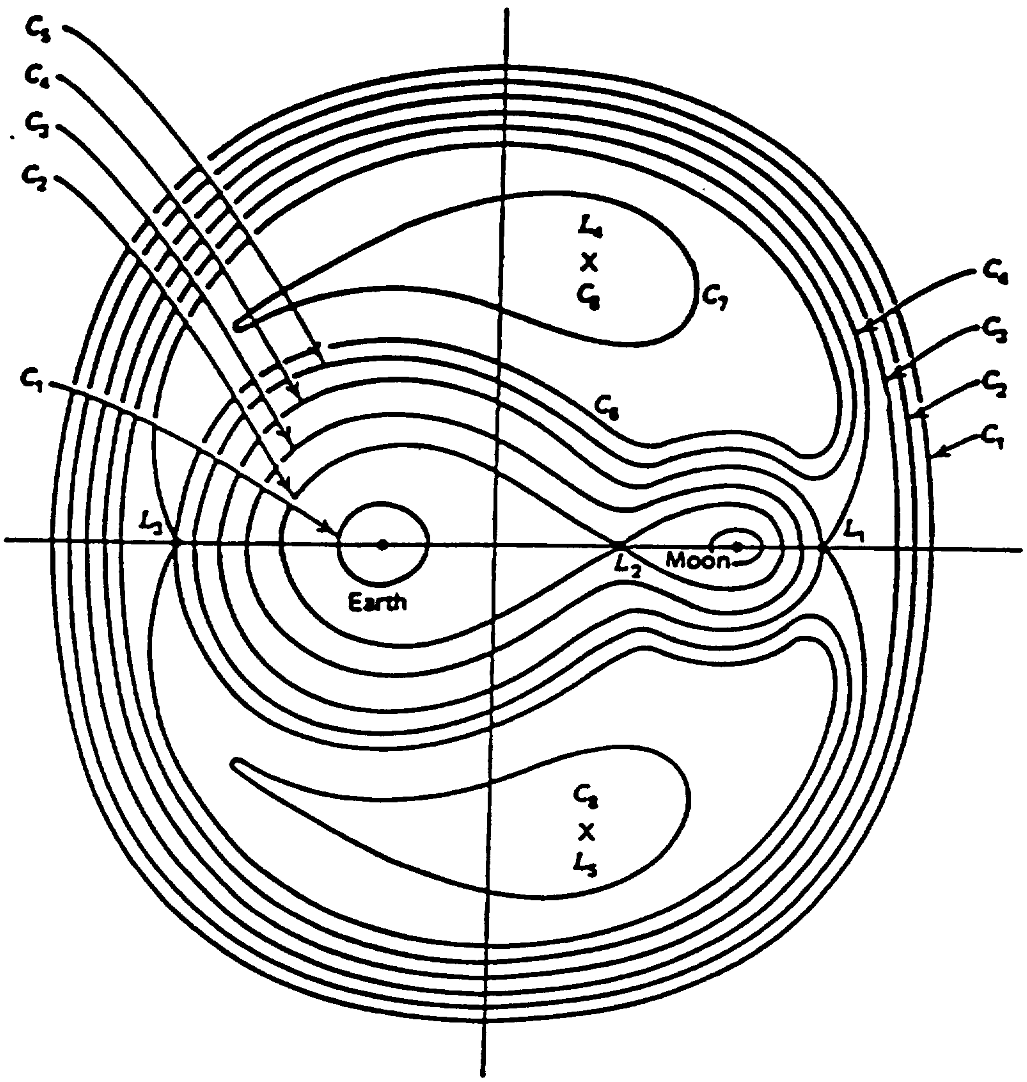


Figure 3.2 Zero Velocity Curves in the Earth-Moon System

(Kaplan, Page 292)

4.0 GUIDANCE METHODOLOGY

The trajectory determination and guidance of a cislunar low-thrust OTV is divided into three distant phases: departure, translunar targeting, and capture. Each of these phases has a different guidance scheme to achieve the overall goal of generating a trajectory between the Earth and the Moon.

4.1 DEPARTURE

The first phase in any cislunar journey for an OTV is the escape from the initial parking orbit, whether about the Earth or the Moon. For low-thrust spacecraft to achieve escape, a long period of continuous thrusting is necessary. This results in a slowly increasing spiral trajectory from the initial orbit. The direction of the thrust vector should be in the direction that has the highest rate of increase of the energy of the orbit per revolution. It can be shown that a near-optimal thrust for an orbital transfer should be directed along the velocity vector of the spacecraft for the majority of the trajectory.[12] This is referred to as tangential thrust, because the thrusting acceleration will be tangent to the trajectory at all times. Another thrusting scheme, circumferential thrust, directs the acceleration along a vector perpendicular to the central body. Tangential thrust is the thrusting approach used in the spiral escape from the departure planet.

4.2 TRANSLUNAR TARGETING

The value of the Jacobian constant of a spacecraft will be used as an indicator of the sufficient energy for the cislunar transfer. Dr. Victor Szebehely[3] notes that an equipotential curve like that shown in Figure 4.1 occurs when the Jacobian constant is approximately 3.3. When the spacecraft is outside of this curve, the range of motion is no longer restricted only to geocentric or lunar orbit, but can include transfer between the neighborhood of the Earth or Moon.

The spacecraft needs to achieve the required Jacobian constant when the velocity vector of the spacecraft is pointed in the appropriate direction to allow transfer between the Earth and the Moon. Figure 4.2 shows this targeting procedure for the OTV. The area about the Earth is divided into four quadrants, I-IV. The Jacobian of the spacecraft is calculated continuously as the spacecraft nears escape. Various values of the Jacobian are chosen experimentally to act as indicators of the spacecraft's proximity to escape. The initial indicator of escape is the value of the Jacobian while the spacecraft is in the third quadrant. If the vehicle achieves a Jacobian of 4.1 while located in the third quadrant, continued thrust will enable a lunar passage to occur. However, if the spacecraft achieves the value of the initial Jacobian, 4.1, while outside the third quadrant, the spacecraft's thrust is turned off. When the spacecraft arrives in the third quadrant the thrust is reinitiated tangentially to obtain the necessary Jacobian for escape and remains on until the spacecraft achieves sufficient energy for transfer and enters the capture phase. On the return trip back from the Moon, the same methodology is used with different Jacobian constants. The Jacobian constants used in the Earth to Moon voyage are driven only by the acceleration level of the spacecraft during escape.

4.3 CAPTURE

As the vehicle approaches the Moon the capture guidance phase of the trajectory is initiated. In the absence of impulsive thrust, the approach and capture to the target body are critical and must not necessitate maneuvering beyond the limited capabilities of the propulsion system. The problem of low-thrust spacecraft guidance and trajectory determination between the Earth and the Moon was addressed in a study by Richard H. Battin and James S. Miller in the late 1950's and early 1960's. The concept for the spacecraft guidance during capture used in this study is derived from Battin and Miller's work.[4]

The guidance scheme is relatively simple and straightforward. The operation of the capture phase guidance is illustrated in Figure 4.3. The velocity of the spacecraft, V_v, relative to the target body (i.e. the Earth or the Moon) is compared with a precalculated velocity, V_c, profile for a spiral capture. This velocity profile is a function of the radial distance from the target body and the magnitude of the thrust acceleration. This velocity difference, V_d, is used in combination with the nominal acceleration to determine the direction and magnitude of the spacecraft thrust during capture.

In order to calculate the velocity as a function of the radial distance from the capturing body, the "ideal", or reference trajectory must be calculated. This is a spiral capture that achieves circular velocity at the desired final altitude. To determine this reference spiral path and the velocity vectors that accompany it, the spacecraft starts in a circular orbit at the desired final altitude about the target body. The mass of the spacecraft at the final altitude is determined by

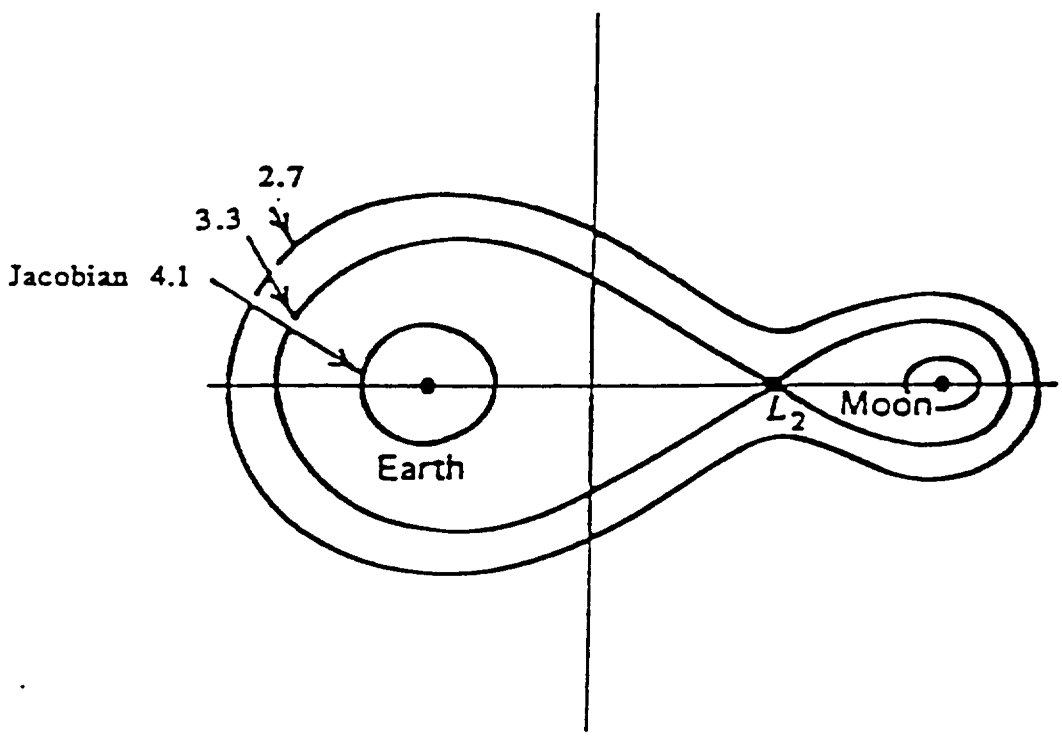


Figure 4.1 - Targeting Velocity Curves

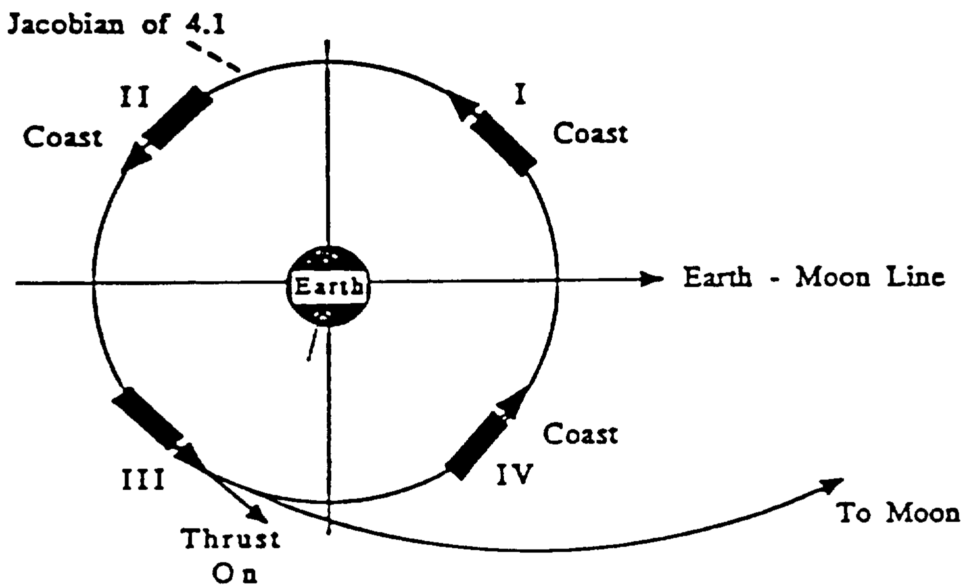


Figure 4.2 - Translunar Targeting

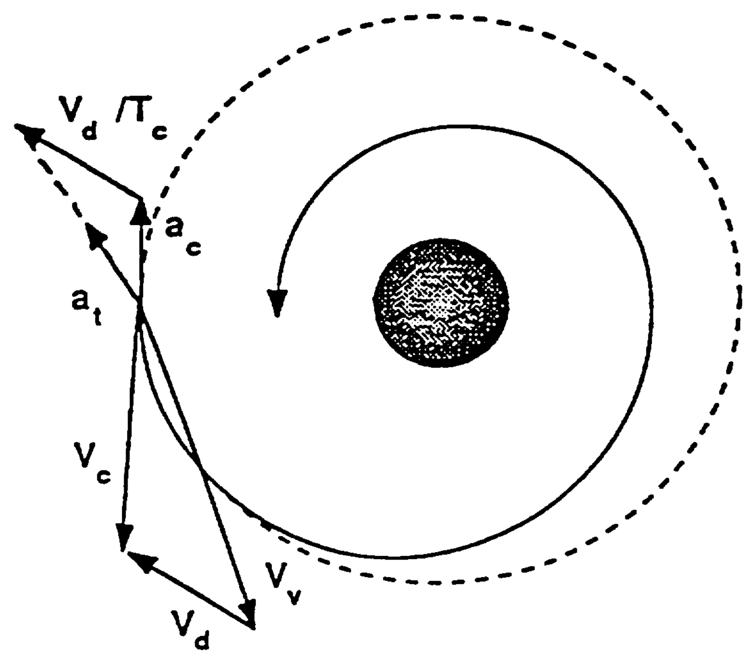


Figure 4.3 - Capture Phase Thrusting Guidance

estimating the final mass of the spacecraft at the completion of the mission to be 80% of the initial mass. The spacecraft follows a spiral out from the target planet using tangentially directed thrust and a negative mass flow. Only the gravitational field of the target planet is considered. The spiral trajectory is otherwise without perturbations and consequently remains two-dimensional. The calculation of the trajectory continues until the energy of the orbit is non-negative, and the vehicle is on a parabolic path. The associated radial and tangential components of the spacecraft's velocity are recorded at select steps as functions of the radial distance from the central body. The velocity functions are obtained by fitting the recorded velocity components to polynomial and power curves. Figures 4.4 and 4.5 are example graphs of the velocity profiles for tangential and radial velocity at a radial distance. The equations shown in the figures have parameterized the velocities as a function of the radial distance. This data was obtained by the described reverse integration process.

An explanation of the thrust guidance control used by the spacecraft during capture phase of the trajectory is presented as follows. The actual velocity of the spacecraft, $\mathbf{V}_v$, at a given radial distance, r, is compared with the parameterized reference capture velocity, $\mathbf{V}_c$, at r. The difference between these velocity vectors is then determined as $\mathbf{V}_d$, where

$$\mathbf{V}_d = \mathbf{V}_v - \mathbf{V}_c \tag{4.1}$$

The instantaneous change in the capture velocity can be approximated as the effect of the acceleration of the spacecraft due to its thrust and the gravitational pull of the planet acting over a small time increment, Δt. This implies

$$\mathbf{V}_c = (\mathbf{a}_c + \mathbf{g})\, \Delta t \tag{4.2}$$

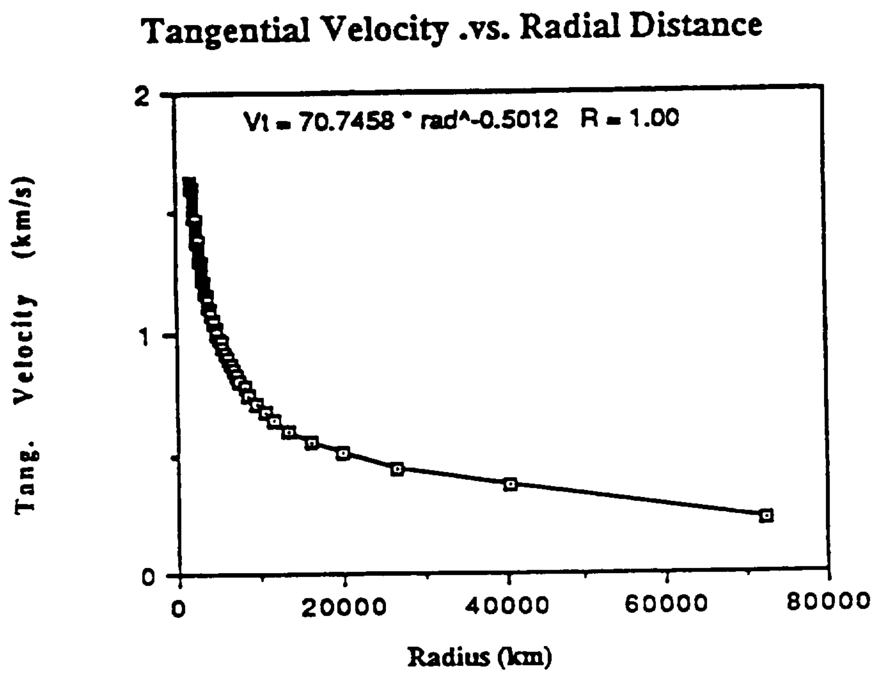


Figure 4.4 Tangential Velocity as a Function of Radial Distance

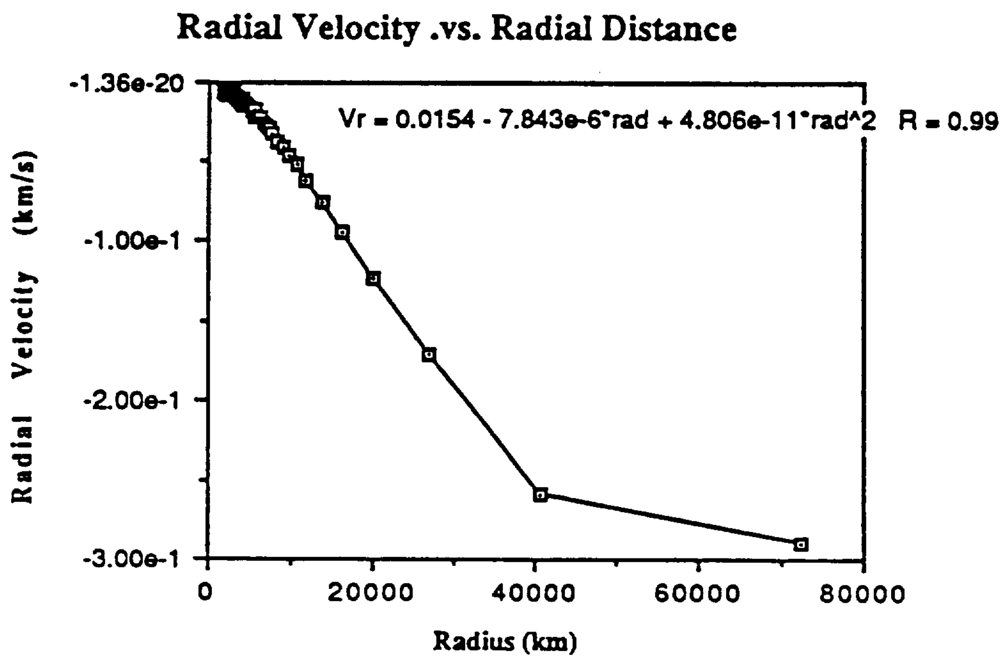


Figure 4.5 Radial Velocity as a Function of Radial Distance

where a_c is the nominal acceleration of the spacecraft, and g is the gravitational acceleration vector of the capture planet. The instantaneous change in the actual velocity of the spacecraft can also be approximated as such

$$\mathbf{V}_v = (\mathbf{a}_t + \mathbf{g})\,\Delta t \tag{4.3}$$

where $\mathbf{a}_t$ is the acceleration vector of the spacecraft on the trajectory. Combining equations 4.1, 4.2, and 4.3 and rearranging to find $\mathbf{a}_t$, equation 4.4 is obtained.

$$\mathbf{a}_t = \mathbf{a}_c - \frac{\Delta \mathbf{V}_d}{\Delta t} \tag{4.4}$$

The thrust acceleration is then chosen so that the rate of change of the velocity vector V_d is proportional to V_d itself. This results in

$$\frac{\Delta \mathbf{V}_d}{\Delta t} = - \frac{\Delta \mathbf{V}_d}{T_c}$$

where T_c is an empirically determined time constant. With this formula the appropriate thrust acceleration can be determined in both magnitude and direction simply with the knowledge of the vehicle's position, velocity, and nominal thrust acceleration $\mathbf{a}_c$.

In the application of the guidance it is reasonable to assume that the direction of the thrust acceleration can be varied at will, but the magnitude of the thrust is limited by the capabilities of the propulsion system. The spacecraft thrust is never required to deliver greater than the nominal thrust. The possibility of a reduction in the thrusting acceleration is not precluded as a desirable effect of the thrusting algorithm. Figure 4.6 is a graphical representation of the acceleration vectors $\mathbf{a}_t$ and $\mathbf{a}_c$. The radii of the circles are determined by the nominal acceleration of the spacecraft.

Then,

$$\mathbf{a}_t \leq \mathbf{a}_c + \frac{\mathbf{V}_d}{\mathbf{T}_c}$$

When the magnitude of the thrust acceleration, a_t, is less than the nominal capabilities of the engine, a less than nominal thrust is needed in the appropriate direction.

This thrusting algorithm drives the spacecraft's velocity components toward the reference velocity conditions. The actual reference conditions can never be reached due to the fact that they were generated in an ideal two-body environment with estimated final conditions, and most importantly, the magnitude of the spacecraft's thrust is limited. This results in a generated spiral capture trajectory that is not "ideal" but is adequate.

---

1. Keaton, Paul W., page 4
2. Hill, P.G., and Peterson, C.R., Chapter 10.
3. Szebehely, Victor, Personal Communication, November, 1987.
4. Battin, R.H., and Miller, J.S.

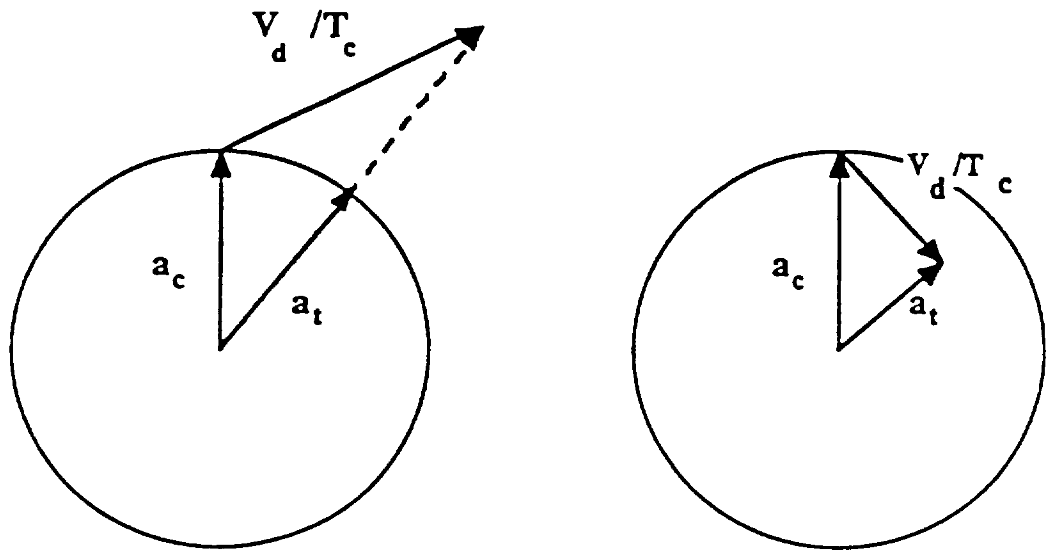


Figure 4.6 - Thrust Guidance Algorithm

5.0 PROGRAM INPUTS

Cislunar needs some preliminary data before it can operate. This data is provided by the user interactively during program execution. A series of screen prompts will direct the user to supply vital information. The user will input the requested data; and the program will proceed to the next question. In this section the screen prompts are discussed as well as the information that the program expects the user to supply.

1. **Prompt:** DO YOU WISH TO SUPPLY S/C CHARACTERISTICS (Y OR N)

   Description: Type "Y" if the default values are to be changed. After hitting RETURN the program will request further information about the spacecraft characteristics. Typing "N" at this prompt instructs the program to use default spacecraft characteristics.

1A. **Prompt:** SPACECRAFT INITIAL MASS =

   Description: Enter the spacecraft mass before it leaves the holding orbit about the planet of origin. The mass is in kilograms. The default value is 60,000 kg.

1B. **Prompt:** SPECIFIC IMPULSE OF ENGINE =

   Description: Enter the engine specific impulse in seconds. Default value is 2500s.

1C. **Prompt:** MASS FLOW RATE OF ENGINES =

Description: Input the propellant mass flow rate. The units are in kilograms per second, and the default value is 0.0082 kg/s.

1D. **Prompt:** DEGREE OF POLYNOMIAL CURVE FIT (2-7)

Description: When requested the program creates a velocity guidance spiral about the target planet. This guidance spiral is used by the control system to provide capture targeting. After creating the data for this guidance spiral, the program forms a curve which approximately "fits" the data. This curve is described with a polynomial mathematical expression. The expression can be between 2nd through 7th order; and the order must be supplied by the user. Note: Higher order curves take longer to create than lower order curves. 3rd order is usually sufficient.

2. **Prompt:** STARTING ORBIT ABOUT THE EARTH OR THE MOON? (E OR M)

Description: Type the first letter of the planetary body from which the spacecraft originates.

3. **Prompt:** DO YOU WANT TO GENERATE PARAMETRIC VELOCITY CURVES?

Description: If the user types "Y" or "Yes" then the program creates the velocity guidance spiral discussed in question 1D. In *FORTRAN* versions of the program, the parametric velocity curves are automatically generated; this question is not asked. In the *BASIC* version of *CISLUNAR*, the user is given the option due to the length of time required to generate these curves. The initial attempts at cislunar targeting are not likely to come close enough to the target planet to make capture guidance worthwhile. In such cases, the velocity guidance spiral generation is not only unnecessary, but also time consuming. If the parametric velocity curves are not desired, the user should type "N" or "No" at this prompt.

4. **Prompt:** INPUT THE RADIUS FROM THE PLANET'S CENTER

Description: Enter the radial distance of the spacecraft from the center of the planet of origin. This distance has units of kilometers.

5. **Prompt:** INPUT ANGLE (DEG.) FROM - X AXIS

Description: The X axis is the line between the Earth and the Moon. The -X axis is the Earth-Moon line on the "Moon" side of the Earth, or on the "Far" side of the Moon. While the +X axis is the Earth-Moon line on the "Far" side of the Earth, or the "Earth" side of the Moon. The program requests that the user supply the angle, measured counter-clockwise, from the -X axis. The angle is in units of degrees.

6. **Prompt:** DO YOU WISH TO SPECIFY THE VELOCITY? (Y OR N)

Description: Enter "N" to tell the program to default to circular orbit speed, otherwise enter "Y".

6A. **Prompt:** INPUT ABSOLUTE VELOCITY

Description: Enter spacecraft velocity in kilometers per second.

7. **Prompt:** MODIFY CONTROL JACOBIANS AND RANGE?

Description: The Jacobian Constant is the controlling parameter for cislunar targeting. The *RANGE* parameter is used to initiate capture guidance and thrust control. The user should type "N" at this prompt if the default values are suitable. Otherwise, enter "Y" and proceed to modify the control Jacobians and the *RANGE*.

7A. **Prompt:** JAC 1 = 4.93

Description: Input the new value for the first Jacobian control point. The Jacobian constant is a non-dimensional number, which can be represented by zero-velocity carves as shown in Figure 4.1. After passing the first control point the spacecraft will thrust only while in control quadrant.

7B. **Prompt:** JAC 2 = 4.10

Description: Enter the new value for the second Jacobian control point. The passage of the second control point means that the thrusting will revert back to continuous mode.

7C. **Prompt:** JAC 3 = 2.70

Description: Enter the new value for the third Jacobian control point. The engines shut down entirely after reaching the third control point. Thrust is zero and the spacecraft coasts.

7D. **Prompt:** *RANGE* = 255000

Description: Input the new *RANGE* in kilometers. The *RANGE* is the distance from the planet of origin at which capture guidance is to begin.

6.0 TEST CASE

An example of a set of inputs for cislunar flight from the Earth which works well is the following:

Prompt: DO YOU WISH TO SUPPLY S/C CHARACTERISTICS? (Y OR N)

Answer: "N"

Prompt: STARTING ORBIT ABOUT THE EARTH OR MOON? (E OR M)

Answer: "E"

Prompt: DO YOU WANT TO GENERATE PARAMETRIC VELOCITY CURVES?

Answer: "Y"

Prompt: INPUT THE RADIUS FROM THE PLANET'S CENTER

Answer: 19000

Prompt: INPUT ANGLE (DEG) FROM -X AXIS

Answer: -76

Prompt: DO YOU WISH TO SPECIFY THE VELOCITY? (Y OR N)

Answer: "N"

Prompt: MODIFY CONTROL JACOBIANS AND RANGE?

Answer: "Y"

Prompt: JAC 1 = 4.93

Answer: 4.93

Prompt: JAC 2 = 4.10

Answer: 4.10

Prompt: JAC 3 = 2.70

Answer: 2.55

Prompt: RANGE = 255000

Answer: 255000

The graphical results are shown in Figures 6.1 and 6.2. The run data is stored in a file called *CISLUNAR.OUT*. This file is not created in BASIC versions since the data is to be presented with the graphics directly on the screen.

The total velocity changes that a vehicle must undergo to perform this mission are derived from Tsiolkovsky's equation (6.1).

Figure 6.1 - CISLUNAR Output: BASIC Version

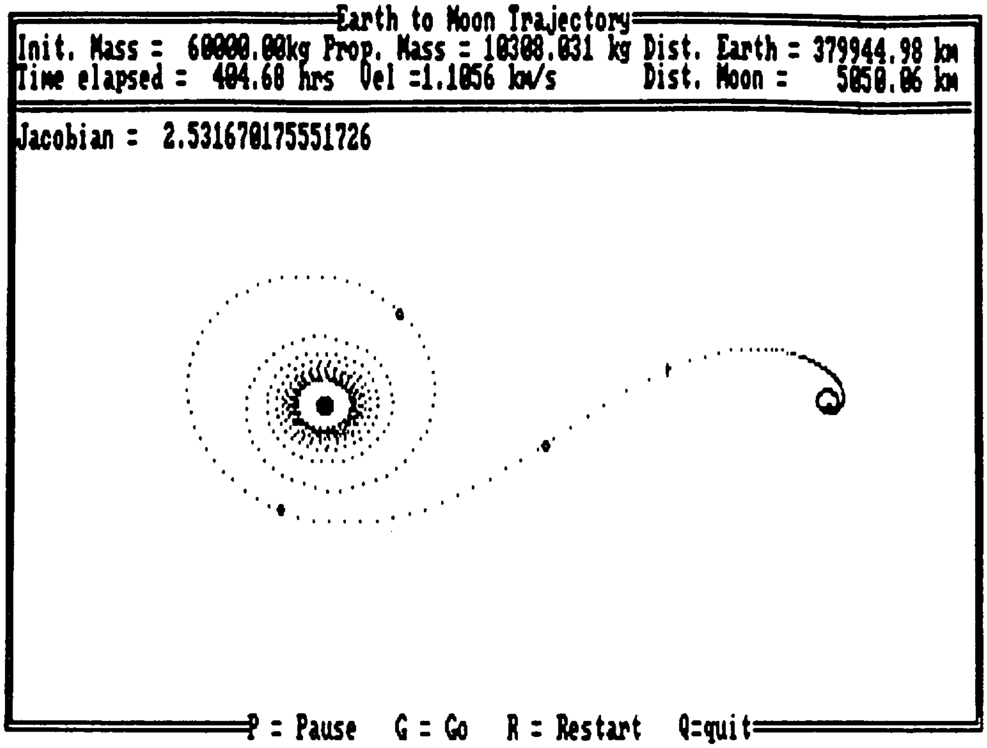
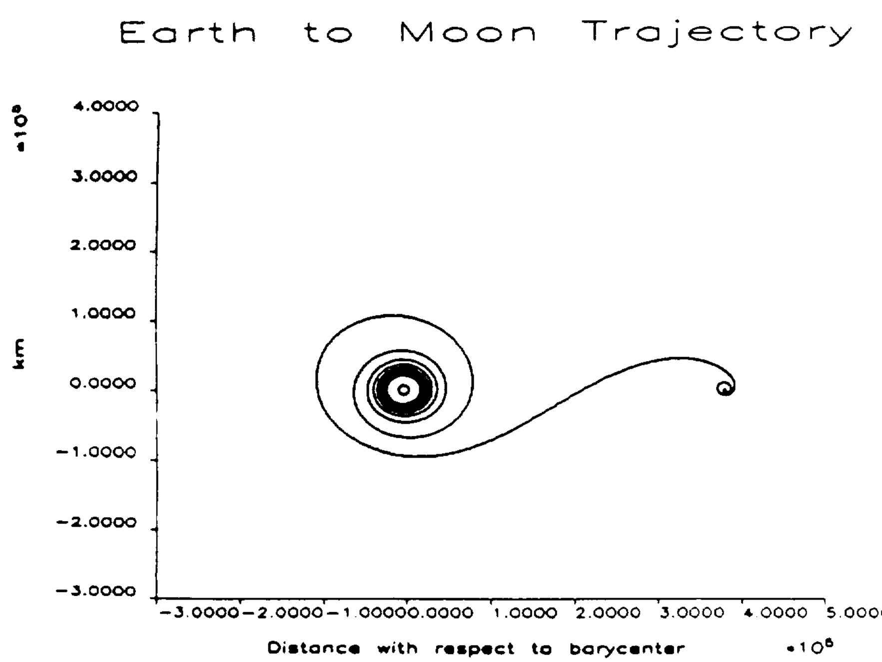


Figure 6.2 - CISLUNAR Output: FORTRAN Version



$$\Delta V = g * I_{sp} * LN \left[\frac{M_o}{M} \right] \tag{6.1}$$

ΔV = Total Velocity Change <km/s>
g = Surface gravity of the Earth (9.81×10^{-3}km/s)
I_{sp} = Spacecraft's Specific Impulse <s>
M_o = Spacecraft's Initial Mass <kg>
M = Spacecraft's Final Mass <kg>

ΔV = 4.619 <km/s>

In order to check the validity of this program, let us compare these results with those that we would receive from other methods of delta-V determination. One method of approximating the total delta-V required for very-low thrust orbital transfers is described in the following paragraph, and will hereafter be called the Approximation of Low-Thrust Velocity Change.

The hypothesis behind the Approximation of Low-Thrust Velocity Change is that the low-thrust delta-V required to transfer between two orbits of a central force body is approximately equal to the difference in their mean orbital speeds. For circular orbits, this is the difference between the circular orbit speeds of the two orbits between which the vehicle is to transfer (Equation 6.2).

$$\Delta V_{\text{Low Thrust}} = \text{Absolute Value of } (V_{m1} - V_{m2}) \tag{6.2}$$

where: V_{m1} = Mean Orbital Velocity of Orbit #1
V_{m2} = Mean Orbital Velocity of Orbit #2

In the test case shown, the spacecraft makes a low-thrust transfer about the Earth between the orbits of 19,000 km circular and the Moon's Orbit of 384,400 km circular. It then makes a low-thrust transfer about the Moon from Escape orbit ($V_m = 0$) to 5,000 km circular. The low-thrust delta-V for these transfers are calculated using Equation 6.2 to be 3.570 <km/s> and 0.990 <km/s> for the Earth and Moon transfers respectively. Therefore, using the Approximation of Low-Thrust Velocity Change, the total low-thrust delta-V is approximately 4.560 <km/s>. This compares well with the 4.619 <km/s> delta-V, obtained using the mass ratio, and Tsiolkovsky's equation.

7.0 PROGRAM CODING

The discussion of the program coding is sectioned by subroutine. The subroutines are addressed in alphabetical order following the discussion of the main "driver" routine. Each subroutine has a description, a list of variables, followed by the actual program code. There are two sets of code. The first set is the code for the BASIC Version of the CISLUNAR; the second set is the FORTRAN Version.

Subroutine Description

And Variable Dictionary

MAIN PROGRAM DESCRIPTION

The main program controls the flow of the entire program. Initially it sets the global constants and the global variables. It calls the IO subroutine which then returns the information necessary to begin the trajectory generation. An integration loop is run to generate the trajectory. While in the loop, the program determines where along the trajectory the spacecraft is and adjusts the integration step size for accuracy and convenience. During the translunar portion of the trajectory the instantaneous Jacobian constant is calculated based on the spacecraft's position and velocity by calling *JACOBI*. The main program compares this Jacobian value with the chosen control variables and marks an indicator along the trajectory when the controls are reached. The spacecraft's mass is decremented according to the integration step size and the mass flow rate, and a new acceleration level for the next pass through the integration loop is calculated. The fourth order Runge-Kutta routine is called to integrate the position and velocity of the spacecraft. The position, velocity, mass, and elapsed time of flight output is updated every fifth integration. In the BASIC Version, each update is sent to the screen. In the FORTRAN Version, each update is stored in two arrays called GRAFX and GRAFY. These arrays are plotted at the end of the simulation. At the end of each integration loop the main program checks to see if there has been input from the keyboard to pause, continue, restart, or quit.

PROGRAM WIDE COMMON VARIABLES

ACCEL1	Acceleration of spacecraft during the spiral reference velocity parametrization portion (km/s^2).
Isp	Specific Impulse of the propulsion system (seconds).
Mdot	Mass flow rate of the propulsion system (kg/s).
MU	Gravitational parameter of the destination planet (km^3/s^2).
RANGE	Range from departure planet that the capture phase is initiated (km).
SCMASS	Initial spacecraft mass (kg).
SCMASSV	Instantaneous spacecraft mass, accounting for the propellant used (kg).
TH	Numeric indicator of the spacecraft's thrust, on (1) or off (0).
THRUST	Thrust of the propulsion system (kg*km/s^2).
VRN	Degree of the radial velocity polynomial curve fit.
VTN	Class of equation for tangential velocity parameterization, linear (1), exponential (2), power (3), or logarithmic (4).

PROGRAM WIDE CONSTANTS

DE	Distance of the Earth's center from the Barycenter of the Earth-Moon system (km).
DM	Distance of the Moon's center from the Barycenter of the Earth-Moon system (km).
EMDIST	Distance between the center of the Earth and the center of the Moon (km).
gravity	Earth's surface gravity (km/s^2).
MUE	Gravitational parameter of the Earth (km^3/s^2).
MUM	Gravitational parameter of the Moon (km^3/s^2).
MUN	Ratio of the Moon's mass to the mass of the Earth-Moon system

NUM	Order of the X state vector for the Runge-Kutta.
NW	Mean motion of the Earth-Moon system (rad/s).
NW2	Mean motion squared (rad/s)^2.
PI	π

ADDITIONAL MAIN PROGRAM VARIABLES

ACCEL	Acceleration of the spacecraft during the cislunar trajectory (km/s^2).
ALPHA	Theta + PI/2, angle of the tangential velocity vector.
CJ	Instantaneous Jacobian constant of the spacecraft.
counter	Counter for determining when to update the screen display of the trajectory.
DIRECTION	Numeric indicator of the direction of the trajectory generation, Earth to Moon (0) or Moon to Earth (1).
dt	Integration step size (seconds).
RANGEOFF	Flag for *GUIDE* subprogram indicating if the spacecraft has entered the capture phase of the trajectory generation.
RJAC()	The array of the Jacobian constants used as controls for the departure portion of the trajectory generation.
RMT	Radial distance of the spacecraft from the center of the Moon (km).
RR	Radial distance of the spacecraft from the center of the Earth (km).
test$	String variable for controlling program from the keyboard.
theta	Angle between the radial vector to the spacecraft from the controlling gravitational body and the x-axis. Dependent upon xf.
thrst	Flag for the spacecraft during the spiral escape indicating the passage of the second control Jacobian, (0) off (1) on.
TIC	Flag for graphically showing the position on the trajectory where the control Jacobians and Range are reached.
TT	Total time of trajectory generation (seconds).

VR() The array of parameterized radial velocity component coefficients.

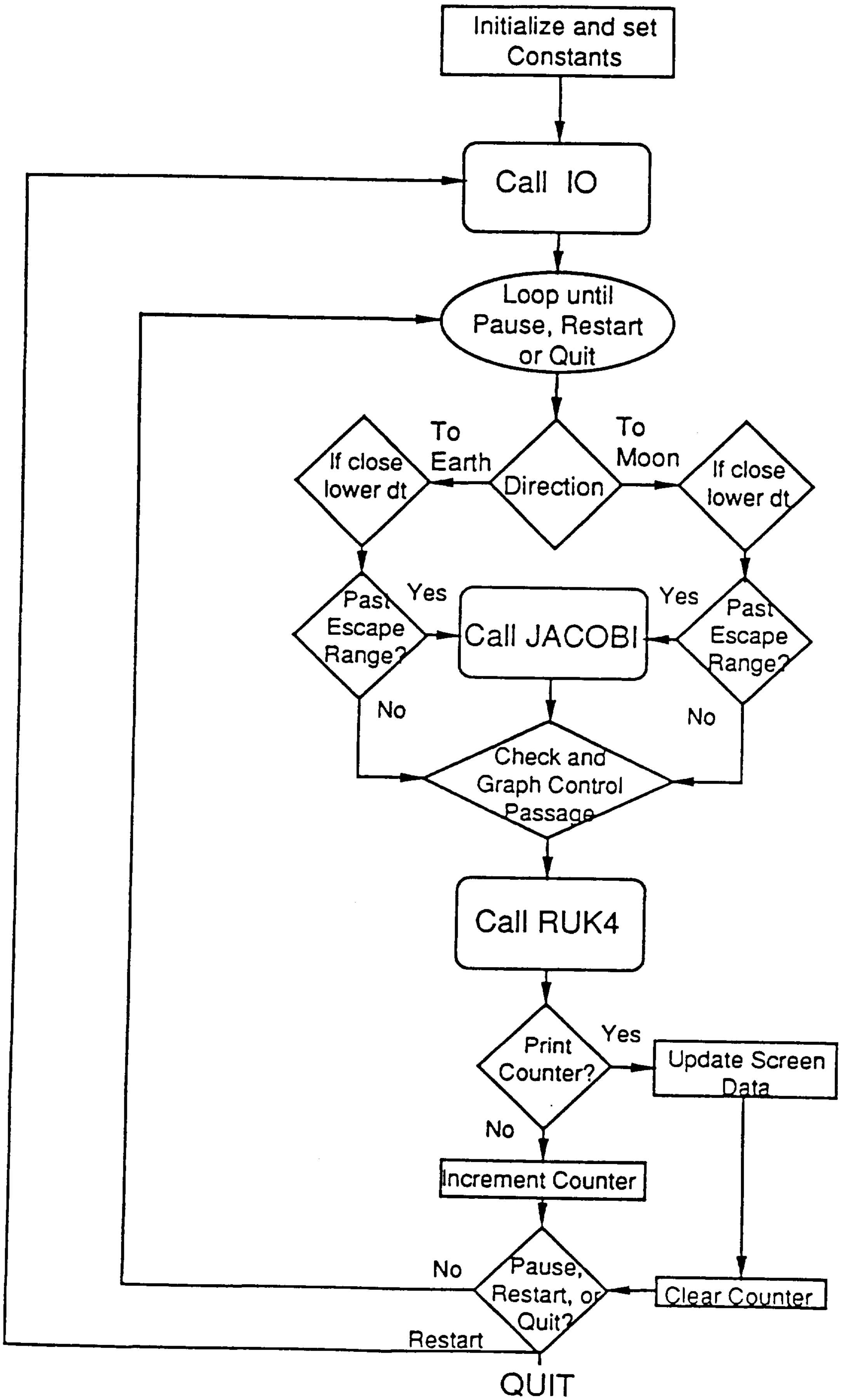
VT() The array of the parameterized tangential velocity component coefficients.

VV Magnitude of the spacecraft's velocity (km/s).

X() State Vector of the spacecraft's position and velocity in the rotating x,y coordinates (km and km/s).

xf Distance along the x-axis the spacecraft is from the Earth or Moon. Dependent upon the phase of the trajectory generation, departure or capture.

Cislunar Program Flowchart



BASIC CODE

```
DECLARE SUB IO (VR#(), VT#(), X#(), DIRECTION#, RJAC#())
DECLARE SUB DER1 (X#(), DX#())
DECLARE SUB RUK (X#(), N!, dt#)
DECLARE SUB SPIRAL (X#(), DIRECTION#, RANGESC#, VT#(), VR#())
DECLARE FUNCTION ACOS# (X#)
DECLARE SUB POLYFIT (PR#(), PV#(), DEGREE#, N#, VR#())
DECLARE SUB CURVE (X#(), Y#(), N#, VT#())
DECLARE SUB DBOX (urow%, ucol%, lrow%, lcol%)
DECLARE FUNCTION ATAN2# (X#, Y#)
DECLARE SUB SHOW (X#())
DECLARE SUB JACOBI (X#(), CJ#)
DECLARE SUB RUK4 (X#(), N!, dt#)
DECLARE SUB DER (X#(), DX#())
DECLARE SUB GUIDE (GD1#, GD2#, X#())
'------------------------------------------------------------------------
Main:        'Main Program for Low-thrust Guidance; 3-body, eqns from Kaplan
'------------------------------------------------------------------------
DEFDBL A-H, K-Z
COMMON SHARED TH, SCMASSV, SCMASS, Isp, THRUST, Mdot, RANGE
COMMON SHARED ACCEL1, MU, VTN, VRN
REM Constants
CONST MUE = 398600.5#                      'mu of the earth
CONST MUM = 4902.794#                      'mu of the moon
CONST DE = 4670.6778#                      'distance of Earth from barycenter (km)
CONST DM = 379729.32#                      'distance of Moon from barycenter  (km)
CONST EMDIST = DE + DM                     'distance between the earth and the moon
CONST NW = .000002665314572#               'mean motion  (rad/s)
CONST NW2 = NW * NW                        'mean motion squared
CONST gravity = 9.809999999999999D-03 'earth's surface gravity (km/s^2)
CONST PI = 3.141592654#                    'pi
CONST NUM = 4!                             'order of state vector for Runge-Kutta
CONST MUN = MUM / (MUM + MUE)              'ratio of Moon's mass to system mass
SCMASS = 60000#                            'total s/c mass (kg)
Isp = 2500#                                'specific impulse (secs)
Mdot = 8.200000000000001D-03               'mass flow rate (kg/s)
VRN = 7                                    'degree of polynomial curve fit
DIM X(4), VR(0 TO 7), VT(2), RJAC(3)
AGAIN:                                     'program position at a restart
IO VR(), VT(), X(), DIRECTION, RJAC()
SCMASSV = SCMASS
TT = 0#:  counter = 5: thrst = 0: TIC = 1: RANGEOFF = 0'initial time
DO
        VV = SQR(X(3) ^ 2 + X(4) ^ 2)
        RR = SQR((X(1) + DE) ^ 2 + X(2) ^ 2)
        RMT = SQR((X(1) - DM) ^ 2 + X(2) ^ 2)
        IF X(1) < 210000 THEN           'earth portion
                xf = X(1) + DE
                dt = INT(.01# * RR)   'integration step size
                IF RR < 7800 AND DIRECTION = 1 THEN dt = 5#
                IF RR > 40000 AND DIRECTION = 0 THEN
                        JACOBI X(), CJ
                        LOCATE 5, 13: PRINT CJ
                ELSE
                        CJ = 10
                END IF
        ELSE                               'lunar portion
                xf = X(1) - DM
                dt = INT(.02# * RMT)  'integration step size
                IF RMT < 2200 AND DIRECTION = 0 THEN dt = 3#
                IF RMT > 6000 AND DIRECTION = 1 THEN
                        JACOBI X(), CJ
                        LOCATE 5, 13: PRINT CJ
                ELSE
                        CJ = 10
                END IF
        END IF
```


ORIGINAL PAGE IS
OF POOR QUALITY

```
        SELECT CASE TIC
        CASE 1: IF CJ < RJAC(TIC) THEN CIRCLE (X(1), X(2)), 2000: TIC = 2
        CASE 2: IF CJ < RJAC(TIC) THEN CIRCLE (X(1), X(2)), 2000: TIC = 3
        CASE 3: IF CJ < RJAC(TIC) THEN CIRCLE (X(1), X(2)), 2000: TIC = 4
        CASE ELSE: TIC = 4
        END SELECT
        theta = ATAN2(xf, X(2))              'angle of radius vector
        ALPHA = theta + PI / 2#              'angle of tangential vector
        TT = TT + dt
        SCMASSV = SCMASSV - Mdot * dt * TH'True s/c mass as propellant is used
        ACCEL = THRUST / SCMASSV             'acceleration of the s/c (km/s^2)
        RUK4 X(), NUM, dt
        IF counter >= 5 THEN
                LOCATE 3, 17: PRINT USING "####.##"; TT / 3600#; : PRINT " hrs"
                LOCATE 3, 35: PRINT USING "#.####"; VV; : PRINT " km/s"
                LOCATE 3, 67: PRINT USING "######.##"; RMT; : PRINT " km"
                LOCATE 2, 67: PRINT USING "######.##"; RR; : PRINT " km"
                LOCATE 2, 40: PRINT USING "#####.###"; SCMASS - SCMASSV; : PRI
 " kg"
                counter = 0
                SHOW X()
        ELSE
                counter = counter + 1
        END IF
        test$ = UCASE$(INKEY$)
        IF test$ = "P" THEN
                DO
                        Test1$ = UCASE$(INKEY$)
                LOOP UNTIL Test1$ = "G"
        END IF
        IF test$ = "R" THEN GOTO AGAIN
LOOP UNTIL test$ = "Q"
CLS
END
```

FORTRAN CODE

```
c=========================================================================
c    Main Program for Low-thrust Guidance; 3-body, eqns from Kaplan
c=========================================================================
C     PROGRAM WRITTEN BY DAVID KORSMEYER
C
C         NASA CONTRACT NAS 17878
C
C         ELECTRIC PROPULSION; TO 87-57; TASK 1.3
C         PROGRAM TRANSLATED AND MODIFIED BY
C          MIKE D'ONOFRIO AND CHRIS VARNER
C          EAGLE ENGINEERING, INC.
C          AUGUST 10,1988
C
      IMPLICIT REAL*16(A-H,K-Z)
      REAL*16 Isp, MALT
      REAL*4  XPLOT,YPLOT,GRAFX,GRAFY
      INTEGER VRN,counter,DIRECT,NUM,AU,TH,thrst,VTN
      CHARACTER*24 TITLED
      DIMENSION X(4), VR(10), VT(5), RJAC(3), GRAFX(20000),
     *        GRAFY(20000), XPLOT(201), YPLOT(201)
C
C   Open Graphics Routines
C
      CALL JBEGIN
      CALL JDINIT ( 1 )
      CALL JDEVON ( 1 )
      CALL JIENAB ( 1, 4, 1 )
C
C   OPEN FILE FOR OUTPUT
C
      OPEN (UNIT = 1, FILE = 'CISLUNAR.OUT', STATUS = 'NEW')
C
C   INITIALIZE VARIABLES
C
      TH = 0
      MUE = 398600.5
      MUM = 4902.794
      DE = 4670.6778
      DM = 379729.32
      EMDIST = DE + DM
      NW=.000002665314572
      NW2 = NW * NW
      gravity = 9.809999999999999D-03
      PI= 3.1415926535
      NUM = 4
      MUN = MUM / (MUM + MUE)
      SCMASS = 60000.0
```

```
      TMAX = 600.0 * 3600.0
      Isp = 2500.0
      Mdot = 8.200000000000001D-03
      ACCEL1 = 0.0
      VRN = 7
      CALL IO (SCMASS, Isp, Mdot, VRN, THRUST, X, gravity,
     + TITLED, RANGE, DIRECT,MUE, VR, rl, theta, RJAC,
     + PI, VT,VRR,MU,ACCEL1,MUM,NUM,DM,DE,dt,STPALT,TT,VTN,VR0)
      SCMASSV = SCMASS
      TT = 0.0
      counter = 5
      thrst = 0
      RANGEOFF = 0
      RR  = QSQRT((X(1) + DE)**2. + X(2)**2. )
      RMT = QSQRT((X(1) - DM)**2. + X(2)**2. )
      PRINT *, '   PLEASE WAIT, GENERATING GRAPH AND DATA FILE'
      DO WHILE( DIRECT .EQ. 0 .AND. RMT .GT. STPALT
     * .OR. DIRECT .EQ. 1 .AND. RR  .GT. STPALT)
        IGRAF = IGRAF + 1
        IF (IGRAPH .GT. 19999) THEN
          PRINT *, 'TOO MANY DATA POINTS'
          STPALT = 1000000.0
          GOTO 200
        ENDIF
        GRAFX(IGRAF) = X(1)
        GRAFY(IGRAF) = X(2)
        VV = QSQRT(X(3)**2. + X(4)**2. )
        RR  = QSQRT((X(1) + DE)**2. + X(2)**2.)
        RMT = QSQRT((X(1) - DM)**2. + X(2)**2.)
        IF (X(1).LT.210000.0) THEN
          xf = X(1)+DE
          dt = INT(.01 * RR)
          IF (RR.LT.7800.0.AND.DIRECT.EQ.1) dt=5.
          IF( RR.GT.40000.0.AND.DIRECT.EQ.0) THEN
            CALL JACOBI(X,CJ,RR,EMDIST,RMT,MUN,VV,NW)
          ELSE
            CJ = 10.
          END IF
        ELSE
C           LUNAR PORTION
          xf = X(1)-DM
          dt = QFLOAT(INT(.02 * RMT))
          IF(RMT.LT.2200.0.AND.DIRECT.EQ.0) dt=3.
          IF(RMT.GT.6000.0.AND.DIRECT.EQ.1) THEN
            CALL JACOBI(X, CJ,RR,EMDIST,RMT,MUN,VV,NW)
          ELSE
            CJ = 10.0
```

```
      END IF
       END IF
       theta = QATAN2( X(2), xf)
       IF (theta .LT. - (PI/2.0) ) theta = theta + 2. * PI
       ALPHA = theta + PI / 2.0
        TT = TT + dt
        SCMASSV = SCMASSV - Mdot * dt * QFLOAT ( TH )
        IF ( SCMASS/SCMASSV .LT. 0.0 ) THEN
          STPALT = 10000000.0
          GOTO 200
        ENDIF
        DELT_V= gravity *Isp *QLOG(SCMASS/SCMASSV)
        ACCEL = THRUST / SCMASSV
        CALL RUK4 (X, NUM, dt, MU,ACCEL1,DM,DE,MUM,
     +     MUE,TH,VR,thrst,CJ,ACCEL,RANGE,DIRECT,ALPHA,
     *  RANGEOFF,theta,RJAC,VTN,DIST,VRR,VRN,VR0,TT,VT)
        IF (counter.GE.5) THEN
          AU = 1
          WRITE(AU,17) TT / 3600.0,VV,RR
 17        FORMAT ('0Time Elapsed = ',F8.2,' hrs.',/,' Velocity    = ',
     +  F9.4, 'km/s',/,' Dist. Earth  = ',F12.2,'km')
          WRITE (AU,171) RMT, SCMASS-SCMASSV, DELT_V
 171       FORMAT (' Dist. Moon   = ',F12.2, 'km',/,' Prop. mass   = ',
     +  F12.2,'kg',/,' Delt vel.    = ',F10.5,'km/s')
          WRITE (AU,172) CJ
 172       FORMAT (' JACOBIAN     = ',E15.8)
          counter=0
        ELSE
          counter = counter + 1
        END IF
 200  END DO
      print *, 'Orbit Completed.'
C
C   DRAW GRAPHICS
C
      CALL GATTRI (1,0,1.0)
      CALL GATTRI (2,0,1.0)
      CALL GATTRI (3,0,1.0)
      CALL GATTRI (4,5,1.0)
      CALL GATTRI (5,5,1.0)
      CALL GATTRI (6,5,1.0)
      CALL GATTRI (7,5,1.0)
      CALL GATTRI (11,5,1.0)
      CALL GATTRI (12,5,1.0)
      CALL GCHART (1,5,TITLED,24)
      CALL GAXIS  (1,0,-250000.0,500000.0,0,'Distance with respect to
     * barycenter ',37,-275000.0,325000.0,0,'km',2)
```

```
      RAD = 6371.23
      DO  38 IPASS =1,2
        IF (IPASS .EQ. 2) RAD = 1739.35
        DO 28 IN = 1, 200
          INM1 = IN - 1
          XPLOT(IN) = RAD * QCOS( QFLOAT (INM1) * 10. * PI/180.)
          YPLOT(IN) = RAD * QSIN( QFLOAT (INM1) * 10. * PI/180.)
          IF ( IPASS .EQ. 1 ) XPLOT ( IN ) = XPLOT ( IN ) - DE
          IF ( IPASS .EQ. 2 ) XPLOT ( IN ) = DM + XPLOT ( IN )
   28     CONTINUE
        CALL JOPEN
        CALL JCOLOR ( 4 )
        CALL GCURVE (XPLOT,YPLOT, 200, 0, 0, 0 )
        CALL JCLOSE
   38   CONTINUE
      CALL JOPEN
      CALL JCOLOR ( 6 )
      CALL GCURVE ( XPLOT, YPLOT, 200, 0, 0, 0 )
      CALL JCOLOR ( 2 )
      CALL GCURVE ( GRAFX, GRAFY, IGRAF, 0, 0, 0 )
      CALL JCLOSE
      CLOSE ( UNIT = 1)
      CALL JPAUSE ( 1 )
      CALL JDEVOF ( 1 )
      CALL JDEND  ( 1 )
      STOP
      END
```

CURVE Subroutine Description

The *CURVE* subroutine is a stand alone linear, logarithmic, power, and exponential curve fitting routine that was taken from a Public Domain library and modified to return the class (i.e. linear, log., power, or exp.) of curve that best fit the input data.

CURVE Subroutine Passed Variables

CALL CURVE (PR(), PVT(), J, VT())

SUB CURVE (X(), Y(), N, VT())

N	The number of data points.
VT()	The array of the coefficients of the curve fit.
X()	The array of the sampled radial positions.
Y()	The array of the tangential velocity component at each sampled position.

BASIC CODE

```
DEFDBL A-H, J-Z
'LEAST MEAN SQUARES CURVE FITTING by Don McDade, Modified by David Korsmeyer
SUB CURVE (X(), Y(), N, VT())
REDIM A(4), B(4), R(4)
' Calculate curves
SX = 0: SY = 0: SXY = 0: SXSQ = 0: SYSQ = 0: SXJ = 0: SYJ = 0: SXYJ = 0
SXJSQ = 0: SYJSQ = 0: SXK = 0: SYK = 0: SXYK = 0: SXKSQ = 0: SYKSQ = 0
SXM = 0: SYM = 0: SXYM = 0: SXMSQ = 0: SYMSQ = 0: J = 0: K = 0: M = 0
FOR I = 1 TO N
SX = SX + X(I): SY = SY + Y(I): SXY = SXY + X(I) * Y(I)
SXSQ = SXSQ + X(I) * X(I): SYSQ = SYSQ + Y(I) * Y(I)    'linear
IF Y(I) > 0 THEN J = J + 1: LY = LOG(Y(I)): SXJ = SXJ + X(I): SYJ = SYJ + LY: SX
YJ = SXYJ + X(I) * LY: SXJSQ = SXJSQ + X(I) * X(I): SYJSQ = SYJSQ + LY * LY'expo
nential
IF X(I) > 0 THEN K = K + 1: LX = LOG(X(I)): SXK = SXK + LX: SYK = SYK + Y(I): SX
YK = SXYK + LX * Y(I): SXKSQ = SXKSQ + LX * LX: SYKSQ = SYKSQ + Y(I) * Y(I)'loga
rithmic
IF X(I) > 0 AND Y(I) > 0 THEN M = M + 1: SXM = SXM + LX: SYM = SYM + LY: SXYM =
SXYM + LX * LY: SXMSQ = SXMSQ + LX * LX: SYMSQ = SYMSQ + LY * LY'power
NEXT I
A(1) = (SY * SXSQ - SX * SXY) / (N * SXSQ - SX * SX)
B(1) = (N * SXY - SX * SY) / (N * SXSQ - SX * SX)
R(1) = (N * SXY - SX * SY) / SQR((N * SXSQ - SX * SX) * (N * SYSQ - SY * SY))
PRINT SPACE$(39): IF B(1) >= 0 THEN PRINT "y="; A(1); "+"; B(1); "x";  ELSE PRIN
T "y="; A(1); B(1); "x";
LOCATE CSRLIN, 56: PRINT "R="; R(1)
IF J < 2 THEN A(2) = 0: B(2) = 0: R(2) = 0
A(2) = EXP((SYJ * SXJSQ - SXJ * SXYJ) / (J * SXJSQ - SXJ * SXJ))
B(2) = (J * SXYJ - SXJ * SYJ) / (J * SXJSQ - SXJ * SXJ)
R(2) = (J * SXYJ - SXJ * SYJ) / SQR((J * SXJSQ - SXJ * SXJ) * (J * SYJSQ - SYJ *
 SYJ))
PRINT "y="; A(2); "e^("; B(2); "x)"; : LOCATE CSRLIN, 56: PRINT "R="; R(2)
IF M < 2 THEN A(3) = 0: B(3) = 0: R(3) = 0
A(3) = EXP((SYM * SXMSQ - SXM * SXYM) / (M * SXMSQ - SXM * SXM))
B(3) = (M * SXYM - SXM * SYM) / (M * SXMSQ - SXM * SXM)
R(3) = (M * SXYM - SXM * SYM) / SQR((M * SXMSQ - SXM * SXM) * (M * SYMSQ - SYM *
 SYM))
PRINT "y="; A(3); "x^("; B(3); ")"; : LOCATE CSRLIN, 56: PRINT "R="; R(3)
IF K < 2 THEN A(4) = 0: B(4) = 0: R(4) = 0
A(4) = (SYK * SXKSQ - SXK * SXYK) / (K * SXKSQ - SXK * SXK)
B(4) = (K * SXYK - SXK * SYK) / (K * SXKSQ - SXK * SXK)
R(4) = (K * SXYK - SXK * SYK) / SQR((K * SXKSQ - SXK * SXK) * (K * SYKSQ - SYK *
 SYK))
IF B(4) >= 0 THEN PRINT "y="; A(4); "+"; B(4); "ln x";  ELSE PRINT "y="; A(4); B
(4); "ln x";
LOCATE CSRLIN, 56: PRINT "R="; R(4)
MAXB = ABS(R(1)): VTN = 1
FOR SORT = 1 TO 4
        MAX = ABS(R(SORT)): CC = SORT
        IF MAX > MAXB THEN MAXB = MAX: VTN = CC
NEXT SORT
VT(1) = A(VTN): VT(2) = B(VTN)
PRINT "NUMBER"; VTN; "WAS CHOSEN"
END SUB
```

FORTRAN CODE

```
c   SQUARES CURVE FITTING by Don McDade, Modified by David Korsmeyer
    SUBROUTINE CURVE (X, Y, N, VT,VR0,VTN)
    IMPLICIT REAL*16 (A-H,O-Z)
    IMPLICIT INTEGER (I-N)
    INTEGER VTN,AU,CC
    REAL*16 MAXB,MAX,LX,LY
    DIMENSION A(4), B(4), R(4), VT(5), X(2100), Y(2100)
    AU = 1
c   Calculate curves
    SX = 0.0
    SY = 0.0
    SXY = 0.0
    SXSQ = 0.0
    SYSQ = 0.0
    SXJ = 0.0
    SYJ = 0.0
    SXYJ = 0.0
    SXJSQ = 0.0
    SYJSQ = 0.0
    SXK = 0.0
    SYK = 0.0
    SXYK = 0.0
    SXKSQ = 0.0
    SYKSQ = 0.0
    SXM = 0.0
    SYM = 0.0
    SXYM = 0.0
    SXMSQ = 0.0
    SYMSQ = 0.0
    J = 0
    K = 0
    M = 0
    DO 10 I = 1, N
     SX = SX + X(I)
     SY = SY + Y(I)
     SXY = SXY + X(I) * Y(I)
     SXSQ = SXSQ + X(I) * X(I)
     SYSQ = SYSQ + Y(I) * Y(I)
     IF( Y(I).GT. 0.0) THEN
      J=J + 1
      LY = LOG(Y(I))
      SXJ = SXJ + X(I)
      SYJ = SYJ + LY
      SXYJ = SXYJ + X(I) * LY
      SXJSQ = SXJSQ + X(I) * X(I)
      SYJSQ = SYJSQ + LY * LY
     END IF
```

```
      IF (X(I).GT.0.0) THEN
       K = K + 1
       LX = LOG(X(I))
       SXK = SXK + LX
       SYK = SYK + Y(I)
       SXYK = SXYK + LX * Y(I)
       SXKSQ = SXKSQ + LX * LX
       SYKSQ = SYKSQ + Y(I) * Y(I)
      END IF
      IF( X(I).GT.0.0. AND. Y(I).GT.0.0) THEN
       M = M + 1
       SXM = SXM + LX
       SYM = SYM + LY
       SXYM = SXYM + LX * LY
       SXMSQ = SXMSQ + LX * LX
       SYMSQ = SYMSQ + LY * LY
      END IF
   10 CONTINUE
    A(1) = (SY * SXSQ - SX * SXY) / (QFLOAT(N) * SXSQ - SX * SX)
    B(1) = (QFLOAT(N) * SXY - SX * SY) / (QFLOAT(N) * SXSQ - SX * SX)
    R(1) = (QFLOAT(N) * SXY - SX * SY) / QSQRT((QFLOAT(N) *
   +SXSQ - SX * SX) * (N * SYSQ - SY * SY))
    IF ( B(1) .GE. 0.0 )  THEN
     WRITE(AU,17) A(1), B(1), R(1)
    ELSE
     WRITE(AU,27) A(1), B(1), R(1)
    END IF
 17 FORMAT('0Y=',E15.8,'+',E15.8,' X',T50, 'R = ',E15.8)
 27 FORMAT('0Y=',2E15.8,' X',T50, 'R = ' , E15.8)
    IF (J.LT.2) THEN
     A(2) = 0.0
     B(2) = 0.0
     R(2) = 0.0
    END IF
    A(2) = EXP((SYJ * SXJSQ - SXJ * SXYJ) /
   +(QFLOAT(J) * SXJSQ - SXJ * SXJ))
    B(2) = (QFLOAT(J) * SXYJ - SXJ * SYJ) /
   + (QFLOAT(J) * SXJSQ-SXJ**2.)
    R(2) = (QFLOAT(J) * SXYJ - SXJ * SYJ) / QSQRT((QFLOAT(J) *
   +SXJSQ - SXJ * SXJ) * (QFLOAT(J) * SYJSQ - SYJ * SYJ))
    WRITE(AU,37) A(2), B(2), R(2)
 37 FORMAT(' y=', E15.8, 'e      (', E15.8, 'x)',T50,'R = ',E15.8)
    IF (M.LT.2) THEN
     A(3) = 0.0
     B(3) = 0.0
     R(3) = 0.0
    END IF
```

```
      A(3) = EXP((SYM * SXMSQ - SXM * SXYM) /(M*SXMSQ - SXM * SXM))
      B(3) = (QFLOAT(M) * SXYM - SXM * SYM) /
     +(QFLOAT(M) * SXMSQ - SXM * SXM)
      R(3) = (QFLOAT(M) * SXYM - SXM * SYM) / QSQRT((QFLOAT(M) * SXMSQ -
     +SXM * SXM) * (QFLOAT(M) * SYMSQ - SYM * SYM))
      WRITE(AU,47) A(3), B(3), R(3)
 47   FORMAT(' y=', E15.8, 'x     (',E15.8, ')',T50,'R = ',E15.8 )
      IF (K.LT.2) THEN
       A(4) = 0.0
       B(4) = 0.0
       R(4) = 0.0
      END IF
      A(4) = (SYK *SXKSQ - SXK *SXYK)/(QFLOAT(K) * SXKSQ -SXK * SXK)
      B(4) = (QFLOAT(K) * SXYK -SXK * SYK) / (QFLOAT(K) * SXKSQ -
     +SXK * SXK)
      R(4) = (QFLOAT(K) * SXYK -SXK * SYK)/QSQRT((QFLOAT(K) *SXKSQ -
     +SXK * SXK) * (QFLOAT(K) * SYKSQ - SYK * SYK))
      IF (B(4).GE.0.0) THEN
       WRITE(AU,57) A(4), B(4), R(4)
 57    FORMAT(' y=',E15.8, '+',E15.8, 'ln x',T50, 'R = ',E15.8)
      ELSE
       WRITE(AU,67) A(4), B(4), R(4)
 67    FORMAT(' y=',E15.8,E15.8, 'ln x',T50,'R = ',E15.8 )
      ENDIF
       MAXB = QABS(R(1))
       VTN = 1
       DO 20 ISORT = 1,4
       MAX = QABS(R(ISORT))
        CC = ISORT
       IF (MAX.GT. MAXB) THEN
         MAXB = MAX
         VTN = CC
        END IF
 20     CONTINUE
       VT(1) = A(VTN)
       VT(2) = B(VTN)
       WRITE(AU,117) VTN
 117    FORMAT ('0NUMBER ', I4 , ' WAS CHOSEN')
       RETURN
       END
```

DER1 Subroutine Description

The DER1 subroutine is called by the Runge-Kutta integration routine from SPIRAL. It contains the equations of motion for a two-body system. The only perturbation force is that of the spacecraft's engine thrust. This thrust is incorporated in the equations of motion in the form of accelerations, a_x and a_y.

$$\ddot{x} = \frac{\mu}{r^2} + a_x$$

$$\ddot{y} = \frac{\mu}{r^2} + a_y$$

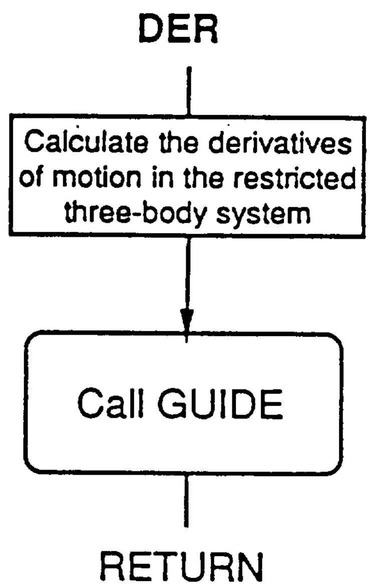
The acceleration of the spacecraft due to the thrust of the propulsion system was determined by calling the subroutine GUIDE. This subroutine returned the values of GD1 and GD2 which are the magnitude of the thrusting acceleration in the x and y direction.

DER1 Subroutine Internal and Share Variables

SUB DER1 (X(), DX()) STATIC

DX()	The array of the derivatives being integrated by the Runge-Kutta.
gmr	The gravitational force on the spacecraft at the given radial distance.
R	Radial distance of the spacecraft from the central body (km).
r2	Square of the radius from the central body (km^2).
V	The magnitude of the velocity of the spacecraft (km/s).
X()	State Vector of the spacecraft's position and velocity (km and km/s).

DER Subroutine Flowchart



BASIC CODE

```
DEFDBL A-H, J-Z
SUB DER1 (X(), DX()) STATIC
DX(1) = X(3)
DX(2) = X(4)
r2 = X(1) ^ 2 + X(2) ^ 2
R = SQR(r2)
gmr = -MU / r2
V = SQR(X(3) ^ 2 + X(4) ^ 2)
DX(3) = gmr * X(1) / R + ACCEL1 * X(3) / V
DX(4) = gmr * X(2) / R + ACCEL1 * X(4) / V
END SUB
```

FORTRAN CODE

```
SUBROUTINE DER1 (X, DX,MU, ACCEL1)
IMPLICIT REAL*16 (A-Z)
DIMENSION X(4),DX(4)
DX(1) = X(3)
DX(2) = X(4)
r2 = X(1) ** 2 + X(2) ** 2
R = QSQRT(r2)
gmr = -MU / r2
V = QSQRT(X(3) ** 2 + X(4) ** 2)
DX(3) = gmr * X(1) / R +ACCEL1 *X(3) / V
DX(4) = gmr * X(2) / R + ACCEL1 * X(4) / V
RETURN
END
```

DER Subroutine Description

The *DER* subroutine contains the equations of motion for the spacecraft in the restricted three-body system. These are:

$$\ddot{x} - n\dot{y} - n^2 x = \frac{\delta}{\delta x}\left[\frac{\mu_{\bullet}}{r_{\bullet}} + \frac{\mu_m}{r_m} \right]$$

$$\ddot{y} + 2n\dot{x} - n^2 y = \frac{\delta}{\delta y}\left[\frac{\mu_{\bullet}}{r_{\bullet}} + \frac{\mu_m}{r_m} \right]$$

Where a_y is the guidance acceleration in the *y*-direction, a_x is the guidance acceleration in the *x*-direction, μ is the gravitational parameter of the target planet, and *r* is the radial distance of the spacecraft from the target planet.

DER Subroutine Internal and Shared Variables

SUB DER (X(), DX()) STATIC

DX()	The array of the derivatives being integrated by the Runge-Kutta.
dx3	The second derivative of the x-component (km/s^2).
dx4	The second derivative of the y-component (km/s^2).
GD1	The x-acceleration of the spacecraft from GUIDE (km/s^2).
GD2	The y-acceleration of the spacecraft from GUIDE (km/s^2).
muroe	The gravitational parameter of the Earth divided by ROE.
murom	The gravitational parameter of the Moon divided by ROM.
ROE	The cubed distance of the spacecraft from the Earth in the rotating x,y coordinates (km^3).
X()	State Vector of the spacecraft's position and velocity in the rotating x,y coordinates (km and km/s).
xde	The x-coordinate of the Earth in the rotating x,y coordinates (km).
xdm	The x-coordinate of the Moon in the rotating x,y coordinates (km).

BASIC CODE

```
DEFDBL A-H, J-Z
SUB DER (X(), DX()) STATIC
DX(1) = X(3)                                'xdot (km/s)
DX(2) = X(4)                                'ydot (km/s)
xde = X(1) + DE: xdm = X(1) - DM            'coordinates of the Earth and Moon
X22 = X(2) ^ 2                              'y-squared
ROE = (xde * xde + X22) ^ 1.5#              'distance of s/c from Earth (km)
ROM = (xdm * xdm + X22) ^ 1.5#              'distance of s/c from Moon  (km)
muroe = MUE / ROE: murom = MUM / ROM 'mu/radius ratios
dx3 = -muroe * xde - murom * xdm + 2# * X(4) * NW + NW2 * X(1) 'x-dbldot (km/s^2
)
dx4 = -X(2) * (muroe + murom) - 2# * X(3) * NW + NW2 * X(2)    'y-dbldot (km/s^2
)
CALL GUIDE(GD1, GD2, X())
DX(3) = dx3 + GD1: DX(4) = dx4 + GD2
END SUB
```

FORTRAN CODE

```
      SUBROUTINE DER (X, DX, DM, DE ,MUM , MUE ,TH ,VR, thrst,
     *          CJ,ACCEL,RANGE,DIRECT,ALPHA,RANGEOFF,THETA,
     *          RJAC,VTN,VT,DIST,VRR,VRN,VR0,TT)
      IMPLICIT REAL*16 (A-H,J-Z)
      DIMENSION DX(4),X(4),VR(10),VT(5), RJAC(3)
      INTEGER VTN,VRN,DIRECT,RANGEOFF,TH,thrst
      NW= 2.665314572E-06
      nw2 = nw**2.
      DX(1) = X(3)
      DX(2) = X(4)
      xde = X(1) + DE
      xdm = X(1) - DM
      X22 = X(2) ** 2.
      ROE = (xde * xde + X22) ** 1.5
      ROM = (xdm * xdm + X22) ** 1.5
      muroe = MUE / ROE
      murom = MUM / ROM
      dx3 = -muroe * xde - murom *xdm +2 * X(4) * NW + NW2 * X(1)
      dx4 = -X(2) * (muroe + murom) - 2 * X(3) * NW + NW2 * X(2)
      CALL GUIDE(GD1, GD2, X, DIRECT,PI,ACCEL,DM,ALPHA,DE,RANGEOFF,
     +      RANGE,thrst,CJ,theta,RJAC,TH,VTN,VT,DIST,VR,VRR,
     +      VRN,VR0,TT)
      DX(3) = dx3 + GD1
      DX(4) = dx4 + GD2
      RETURN
      END
```

ERASE Subroutine Description

This subroutine is used to reinitialize the velocity component arrays. The arrays are used to provide curve fit data to the POLYFIT subroutine. This subroutine does not exist in the BASIC version of CISGRAPH.

ERASE Subroutine Variables

VRO First point of the radial velocity array used to produce guidance curves.

VR() Radial velocity array used to produce guidance curves.

VT() Tangential velocity array used to produce guidance curves.

FORTRAN CODE

```
      SUBROUTINE ERASE(VR, VT, VR0)
      REAL*16  VR0,VR(10),VT(5)
      VR0= 0.0
      DO 10 I=1,5
       VR(I) = 0.0
       VR(I+5) = 0.0
       VT(I) = 0.0
10    CONTINUE
      RETURN
      END
```

GUIDE Subroutine Description

The *GUIDE* subroutine controls the direction and magnitude of the spacecraft's thrusting acceleration. A set of guidance parameters are determined depending on the direction of the spacecraft's trajectory. When the spacecraft is in the departure phase of the trajectory the acceleration is along the velocity vector of the spacecraft. When the spacecraft Jacobian value passes the first control Jacobian, Jac1, the program checks the quadrant the spacecraft is in, and leaves the thrust on, if it is in the appropriate quadrant; turns it off if not. If the spacecraft's Jacobian value has passed the second control Jacobian, Jac2, then the spacecraft returns to continuously thrusting along the velocity vector. When the spacecraft's Jacobian has passed the third control Jacobian, Jac3, the spacecraft's thrust is turned off and the spacecraft drifts until its distance from the departure planet has passed the value of RANGE. This initiates the capture phase of the trajectory. The tangential and radial components of the reference trajectory are calculated from their parametric functions and the direction and magnitude of the capture acceleration is determined.

GUIDE Subroutine Internal and Shared Variables

SUB GUIDE (GD1, GD2, X())

ACCEL	Acceleration of the spacecraft during the cislunar trajectory (km/s^2).
ACXT	The tangential acceleration of the spacecraft in the x-direction (km/s^2).
ACYT	The tangential acceleration of the spacecraft in the y-direction (km/s^2).
alph	The angle between the velocity vector and the tangential component of velocity (radians).
ALPHA	theta + PI/2
am	Magnitude of the capture guidance acceleration (km/s^2).
amx	The capture guidance acceleration in the x-direction (km/s^2).
amy	The capture guidance acceleration in the y-direction (km/s^2).
ANGLE1	The angle defining the beginning of the third quadrant area around the departure planet during the departure phase of the trajectory generation (radians).
ANGLE2	The angle defining the end of the third quadrant area around the departure planet during the departure phase of the trajectory generation (radians).
CJ	Instantaneous Jacobian constant of the spacecraft.
DIRECTION	Numeric indicator of the direction of the trajectory generation, Earth to Moon (0) or Moon to Earth (1).
DIST	The distance the spacecraft is away from the capture planet (km).
dt	Integration step size (seconds).
GD1	The x-acceleration of the spacecraft from the guidance control equations (km/s^2).
GD2	The y-acceleration of the spacecraft from the guidance control equations (km/s^2).
r1	The distance from the capture planet (km/1000).
RANGE	Range from initial planet that the capture phase is initiated (km).

RANGEOFF Flag for GUIDE subprogram indicating if the spacecraft has entered the capture phase of the trajectory generation.

RJAC() The array of the Jacobian constants used as controls for the departure portion of the trajectory generation.

tc Empirical time constant used to change the control velocity difference into an acceleration (seconds).

TH Numeric indicator of the spacecraft's thrust, on (1) or off (0).

theta Angle between the radius vector to the spacecraft from the controlling gravitational body and the x-axis. Dependent upon xf.

thrst Flag for the spacecraft on its spiral escape indicating the passage of the second control Jacobian, (0) off (1) on.

TT Total time of trajectory generation (seconds).

VMAG Magnitude of the velocity of the spacecraft (km/s^2).

VR() The array of parameterized radial velocity component coefficients.

VRN Degree of the radial velocity polynomial curve fit.

VRR The parameterized radial velocity magnitude (km/s).

VT() The array of the parameterized tangential velocity component coefficients.

VTN Class of equation for tangential velocity parameterization, linear (1), exponential (2), power (3), or logarithmic (4).

VTT The parameterized tangential velocity magnitude (km/s).

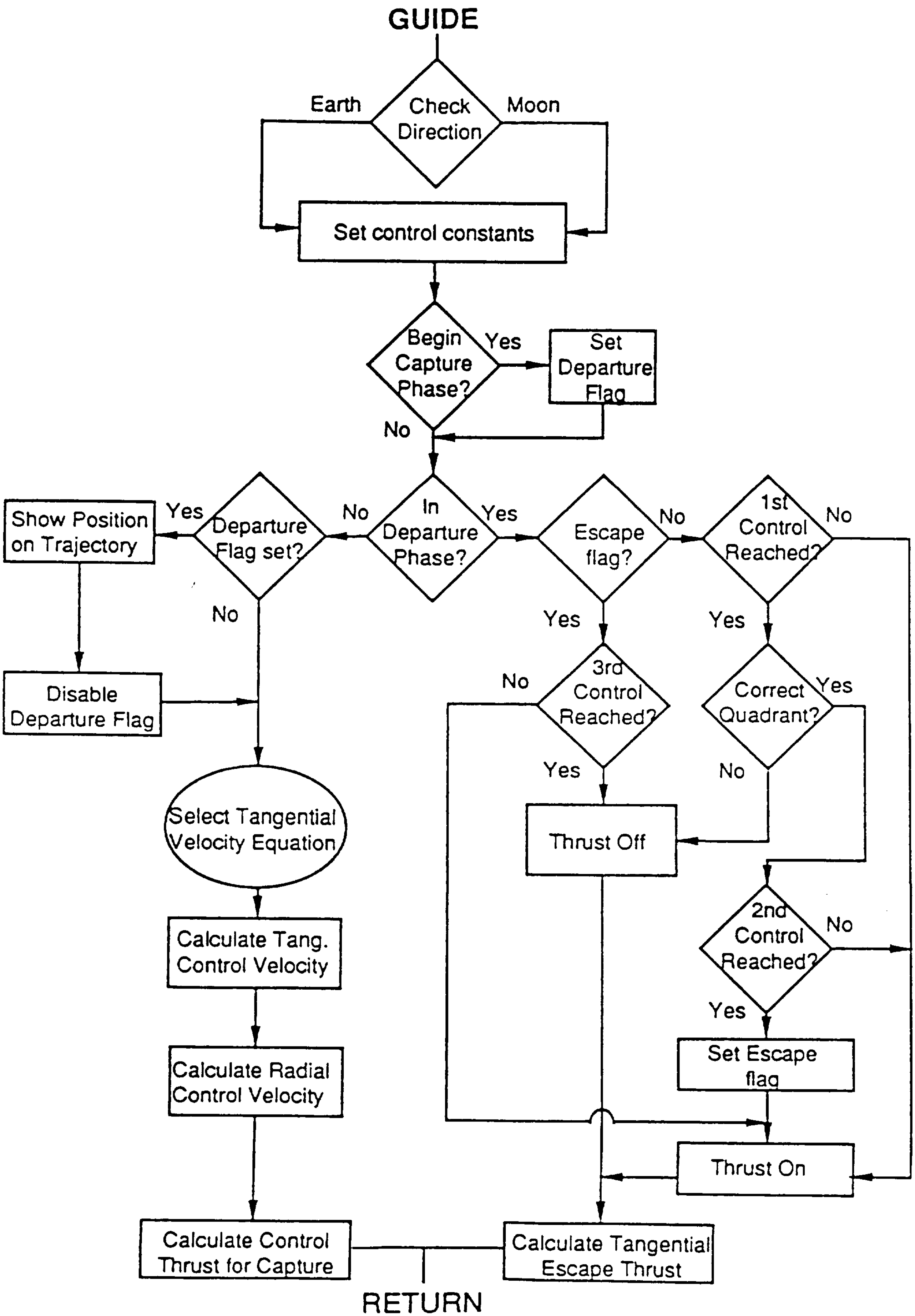
VTV Loop variable for the radial velocity polynomial.

X() State Vector of the spacecraft's position and velocity in the rotating x,y coordinates (km and km/s).

XACCEL The magnitude (+ or -) of the ACCEL used (km/s^2).

XRANGE The magnitude of the distance in the x - direction the spacecraft is from the departure planet (km).

GUIDE Subroutine Flowchart



BASIC CODE

```
DEFDBL A-H, J-Z
SUB GUIDE (GD1, GD2, X())                              'Guidance subprogram
SHARED TT, theta, ALPHA, CJ, dt, SCMASSV, thrst
SHARED VT(), VR(), DIRECTION, ACCEL, RJAC(), RANGEOFF
SELECT CASE DIRECTION
        CASE 0                                    'earth to moon
        ANGLE1 = PI: ANGLE2 = 1.5# * PI: XACCEL = ACCEL
        DIST = SQR((X(1) - DM) ^ 2 + X(2) ^ 2): XRANGE = X(1): alph = ALPHA
        CASE 1                                    'moon to earth
        ANGLE1 = -6#: ANGLE2 = 0#: XACCEL = -ACCEL
        DIST = SQR((X(1) + DE) ^ 2 + X(2) ^ 2): XRANGE = ABS(DM - X(1)): alph
ALPHA + PI
END SELECT
VMAG = SQR(X(3) ^ 2 + X(4) ^ 2)
ACXT = ACCEL * X(3) / VMAG: ACYT = ACCEL * X(4) / VMAG 'components of accel, T
gential
IF RANGEOFF = 1 THEN RANGE = 50000
IF XRANGE < RANGE THEN
   IF thrst = 0 THEN
        IF CJ < RJAC(1) THEN
                IF theta > ANGLE1 AND theta < ANGLE2 THEN
                        IF CJ < RJAC(2) THEN thrst = 1
                        TH = 1#
                ELSE
                        TH = 0#
                END IF
        ELSE
        TH = 1#
        END IF
   ELSEIF CJ > RJAC(3) THEN
        TH = 1#
        ELSE
        TH = 0#
   END IF
   GD1 = ACXT * TH: GD2 = ACYT * TH 'thrusting tangentially spiral out
ELSE
        IF RANGEOFF = 0 THEN LINE (X(1), X(2) + 5000)-(X(1), X(2) - 5000), 15
        RANGEOFF = 1
        tc = 100#: rl = DIST / 1000#
        SELECT CASE VTN
        CASE 1
                VTT = VT(1) + VT(2) * rl
        CASE 2
                VTT = VT(1) * EXP(VT(2))
        CASE 3
                VTT = VT(1) * (rl) ^ VT(2)
        CASE 4
                VTT = VT(1) + VT(2) * LOG(rl)
        END SELECT
        VRR = VR(0)
        FOR VTV = 1 TO VRN
                VRR = VRR + VR(VTV) * rl ^ VTV
        NEXT VTV
        amx = -ACXT + ((VRR * COS(theta) - VTT * COS(alph)) - X(3)) / tc
        amy = -ACYT + ((VTT * SIN(alph) + VRR * SIN(theta)) - X(4)) / tc
        am = SQR(amx ^ 2 + amy ^ 2)
        GD1 = ACCEL / am * amx
        GD2 = ACCEL / am * amy: TH = 1#
END IF
END SUB
```

FORTRAN CODE

```
      SUBROUTINE GUIDE (GD1, GD2, X, DIRECT, PI, ACCEL, DM,ALPHA,DE,
     +          RANGEOFF, RANGE, thrst, CJ, theta, RJAC,TH,
     +          VTN,VT,DIST,VR,VRR,VRN,VR0,TT)
      IMPLICIT REAL*16 (A-H,J-Z)
      IMPLICIT INTEGER (I)
      INTEGER VTN,VRN,DIRECT,RANGEOFF,TH,thrst
      DIMENSION X(4),RJAC(3),VT(5),VR(10)
C       IF (TT/3600. .GT. 312. )PRINT *, 'VR0 ', VR0
      PI = 3.1415926535
      IF (DIRECT.EQ.0) THEN
       ANGLE1 = PI
       ANGLE2 = 1.5 * PI
       XACCEL = ACCEL
       DIST = QSQRT((X(1) - DM) ** 2. + X(2) ** 2.)
       XRANGE = X(1)
       alph = ALPHA
      ELSE
       ANGLE1 = -6.0
       ANGLE2 = 0.0
       XACCEL = -ACCEL
       DIST = QSQRT((X(1) + DE) ** 2. + X(2) ** 2.)
       XRANGE = QABS(DM-X(1))
       Alph = ALPHA + PI
      ENDIF
      VMAG = QSQRT(X(3) ** 2. + X(4) ** 2.)
      ACXT = ACCEL * X(3) / VMAG
      ACYT = ACCEL * X(4) / VMAG
      IF (RANGEOFF.EQ.1) RANGE = 50000
      IF (XRANGE.LT.RANGE) THEN
       IF ( thrst .EQ. 0 ) THEN
        IF ( CJ .LT. RJAC(1) ) THEN
         IF ( theta .GT. ANGLE1 .AND. theta .LT. ANGLE2 ) THEN
          IF ( CJ. LT. RJAC(2) ) thrst = 1
          TH=1
         ELSE
          TH=0
         ENDIF
        ELSE
         TH=1
        ENDIF
       ELSE IF (CJ.GT.RJAC(3)) THEN
        TH=1
       ELSE
        TH=0
       ENDIF
       GD1=ACXT * QFLOAT(TH)
       GD2 = ACYT * QFLOAT(TH)
```

```
      ELSE
        RANGEOFF=1
        tc = 100.
        rl = DIST / 1000.
        IF (VTN.EQ.1) VTT=VT(1) +VT(2) * rl
        IF (VTN.EQ.2) VTT=VT(1) * EXP(VT(2))
        IF (VTN.EQ.3) VTT = VT(1) * rl **VT(2)
        VRR = VR0
        IF (VTN.EQ.4) VTT = VT(1) + VT(2) * LOG(rl)
        DO 10 VTV= 1,VRN
        VRR= VRR + VR(VTV) * rl ** VTV
 10      CONTINUE
        amx = -ACXT + ((VRR * QCOS(theta) - VTT*QCOS(alph))-X(3))/tc
        amy = -acyt + ((VTT*QSIN(alph) + VRR*QSIN(theta))-X(4))/tc
        am=QSQRT(amx**2. + amy**2.)
        GD1 = ACCEL/ am * amx
        GD2 = ACCEL/ am * amy
        TH = 1
      ENDIF
      RETURN
      END
```

IO Subroutine Description

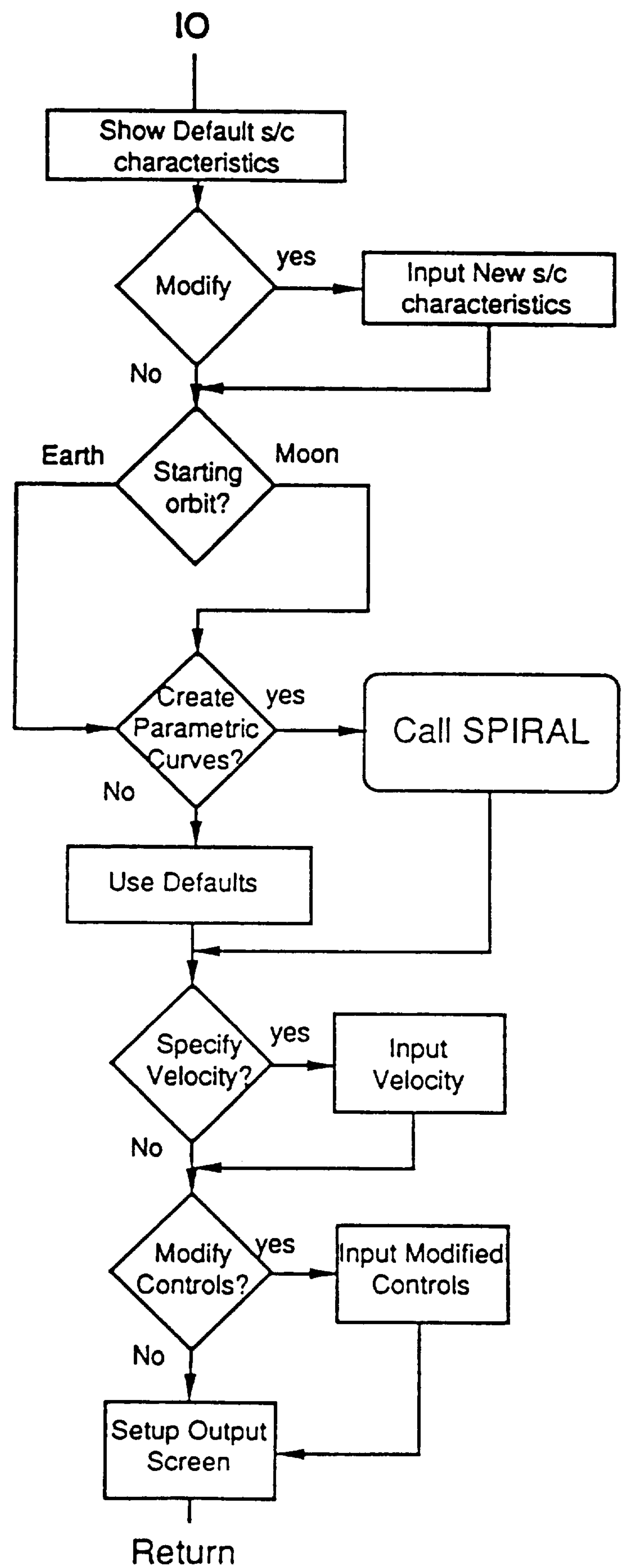
The IO subroutine handles the initial input of spacecraft characteristics, control variables, and various use choices. The initial output screen formatting and graphics setup is also performed in this subroutine. The spacecraft's initial operating characteristics, such as initial mass, I_{sp}, and the mass flow rate of the propulsion system, are selected in addition to the direction of the trajectory generation. Then the user is asked whether a new set of guidance velocity parametrics should be generated based upon the input spacecraft characteristics. If the response is yes, the program control is passed to another subroutine called *SPIRAL*. This subroutine generates the parametric velocity profiles. Upon completion of *SPIRAL* or if the response to generating velocity parametrics is no, the IO subroutine prompts the user to input the spacecraft's initial position and velocity. The next question asks if the user would like to modify the control Jacobians and Range. These control values are used by the program to control when the spacecraft escapes from its outbound spiral and begins the capture phase of the trajectory. The default values are presented and the user can modify any of them. After all of the inputs and responses have to be recorded the IO subroutines sets up the screen for the trajectory generation and returns control to the Main program.

IO Subroutine Internal and Shared Variables

SUB IO (VR(), X(), DIRECTION, RJAC()) STATIC

DIRECTION	Numeric indicator of the direction of the trajectory generation, Earth to Moon (0) or Moon to Earth (1).
r1	Initial radius value, input from keyboard (km).
RJAC()	The array of the Jacobian constants used as controls for the departure portion of the trajectory generation.
thet	Angle, in radians, of theta.
theta	Initial angle, in degrees, spacecraft is from - x-axis, input from keyboard.
title$	String Variable for the title of the output screen.
vexp	Velocity magnitude of spacecraft, input from keyboard (km).
VR()	The array of the parameterized radial velocity component coefficients.
VT()	The array of the parameterized tangential velocity component coefficients.
X()	State Vector of the spacecraft's position and velocity in the rotating x,y coordinates (km and km/s).

IO Subroutine Flowchart



BASIC CODE

```
DEFDBL A-H, K-Z
SUB IO (VR(), VT(), X(), DIRECTION, RJAC()) STATIC
CLS
SCREEN 0
PRINT "ZDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDD?"
PRINT "3   Trajectory Generation Model for Low-Thrust OTVs in Cislunar Space
   3"
PRINT "3   Using a Thrusting Control Algorithm in the Restricted three-body
   3"
PRINT "3   formulation of the Earth-Moon system.  {Gus Babb copy} - DJK, 4/1/88
   3"
PRINT "@DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDY"
DBOX 6, 1, 11, 80
 LOCATE 7, 3: PRINT "Spacecraft intial mass =", SCMASS;
 LOCATE 8, 3: PRINT "Specific Impulse of engine =", Isp;
 LOCATE 9, 3: PRINT "Mass flow rate of engines = ", Mdot
 LOCATE 10, 3: INPUT "Do you wish to specify s/c characteristics? (y or n)", B$
IF UCASE$(B$) = "Y" THEN
 DBOX 11, 1, 16, 80
 LOCATE 12, 3: INPUT "Spacecraft intial mass =", SCMASS
 LOCATE 13, 3: INPUT "Specific Impulse of engine =", Isp
 LOCATE 14, 3: INPUT "Mass flow rate of engines = ", Mdot
 LOCATE 15, 3: INPUT "Degree of polynomial curve fit (2-7) ", VRN
END IF
THRUST = gravity * Isp * Mdot          'thrust dependent on mass flow and isp
PRINT
INPUT "Starting Orbit about the Earth or the Moon? (e or m) ", A$
IF UCASE$(A$) = "M" THEN
        DIRECTION = 1                  'flag indicates s/c going moon to earth
        RANGE = 110000
        title$ = "Moon to Earth Trajectory"
        INPUT "Do you want to generate parmetric velocity curves? ", A$
        IF UCASE$(A$) = "Y" THEN
                ERASE VT, VR, X
                X(0) = 0#: X(2) = 7178#: X(3) = -SQR(MUE / X(2)): X(4) = 0#
                SPIRAL X(), DIRECTION, 225000, VT(), VR()
        ELSE
                VR(0) = 10.92448764296168#
                VR(1) = -.6486459794313275#
                VR(2) = .0233450835020553#
                VR(3) = -4.502555723665775D-04
                VR(4) = 4.767029852095007D-06
                VR(5) = -2.760604361514407D-08
                VR(6) = 8.185026033125225D-11
                VR(7) = -9.6764120959764D-14
                VT(1) = 3.880360674664431D-02
                VT(2) = -4.680399272731233D-03
                VTN = 1
        END IF
        INPUT "Input the radius from the Moon's center ", r1
        INPUT "Input angle (deg) from -x axis ", theta
        INPUT "Do you wish to specify the velocity? (y or n) ", B$
        IF UCASE$(B$) = "Y" THEN
                INPUT "Input absolute velocity ", vexp
          ELSE
               vexp = -SQR(MUM / r1)
        END IF
        thet = theta * PI / 180#
        X(1) = r1 * COS(thet) + DM
        X(2) = r1 * SIN(thet)
        X(3) = -vexp * SIN(thet)
        X(4) = vexp * COS(thet)
        INPUT "Modify Control Jacobians and Range? ", C$
        IF UCASE$(C$) = "Y" THEN
```

```
                    PRINT "Default Values 3.05, 2.82, 2.50"
                    PRINT "Jac1 = ", RJAC(1): INPUT RJAC(1)
                    PRINT "Jac2 = ", RJAC(2): INPUT RJAC(2)
                    PRINT "Jac3 = ", RJAC(3): INPUT RJAC(3)
                    PRINT "Range = ", RANGE: INPUT RANGE
            ELSE
                    RJAC(1) = 3.05#: RJAC(2) = 2.82#: RJAC(3) = 2.5#
            END IF
    ELSE
            DIRECTION = 0                     'flag indicates s/c going earth to moon
            RANGE = 255000
            title$ = "Earth to Moon Trajectory"
            INPUT "Do you want to generate parmetric velocity curves? ", A$
            IF UCASE$(A$) = "Y" THEN
                    ERASE VT, VR, X
                    X(0) = 0#: X(2) = 1838#: X(3) = -SQR(MUM / X(2)): X(4) = 0#
                    SPIRAL X(), DIRECTION, 120000, VT(), VR()
            ELSE
                    VR(0) = 2.083801929462969#
                    VR(1) = -.328978346551485#
                    VR(2) = 2.759195072751214D-02
                    VR(3) = -1.13520698564688D-03
                    VR(4) = 2.454296131544368D-05
                    VR(5) = -2.850460877535156D-07
                    VR(6) = 1.681222735220663D-09
                    VR(7) = -3.948910484014757D-12
                    VT(1) = 2.2187#
                    VT(2) = -.5012#
                    VTN = 3
            END IF
            PRINT "Input the radius from the Earth's center "
            INPUT r1: PRINT "Input angle (deg) from -x axis "
            INPUT theta: INPUT "Do you wish to specify the velocity? (y or n) ", B$
            IF UCASE$(B$) = "Y" THEN
                    INPUT "Input absolute velocity ", vexp
              ELSE
                    vexp = SQR(MUE / r1)
            END IF
            thet = theta * PI / 180#
            X(1) = r1 * COS(thet) - DE
            X(2) = r1 * SIN(thet)
            X(3) = -vexp * SIN(thet)
            X(4) = vexp * COS(thet)
            INPUT "Modify Control Jacobians and Range? ", C$
            IF UCASE$(C$) = "Y" THEN
                    PRINT "Default Values 4.93, 4.10, 2.70"
                    PRINT "Jac1 = ", RJAC(1): INPUT RJAC(1)
                    PRINT "Jac2 = ", RJAC(2): INPUT RJAC(2)
                    PRINT "Jac3 = ", RJAC(3): INPUT RJAC(3)
                    PRINT "Range = ", RANGE: INPUT RANGE
            ELSE
                    RJAC(1) = 4.93#: RJAC(2) = 4.1#: RJAC(3) = 2.7#
            END IF
    END IF
    CLS 1
    SCREEN 2: WINDOW (-250000, -275000)-(500000, 325000)
    CIRCLE (-DE, 0), 6378
    PAINT (-DE, 0), 9, 15
    CIRCLE (DM, 0), 1734
    PAINT (DM, 0), 8, 15
    DBOX 1, 1, 4, 80
    DBOX 1, 1, 25, 80
    LOCATE 1, 28: PRINT title$
    LOCATE 2, 2: PRINT "Init. Mass = "; : PRINT USING "######.##"; SCMASS; : PRINT "
    kg"
    LOCATE 2, 27: PRINT "Prop. Mass = ";
```

```
LOCATE 3, 53: PRINT "Dist. Moon = "; : LOCATE 2, 53: PRINT "Dist. Earth = ";
LOCATE 3, 2: PRINT "Time elapsed = ";
LOCATE 3, 30: PRINT "Vel = ";
LOCATE 5, 2: PRINT "Jacobian = ";
LOCATE 25, 21: PRINT "P = Pause   G = Go   R = Restart   Q=quit";
END SUB
```

FORTRAN CODE

```
      SUBROUTINE IO (SCMASS,Isp,Mdot,VRN,THRUST,X,
     * gravity,TITLED,RANGE,DIRECT,MUE,VR,rl,theta,RJAC, PI,
     * VT,VRR,MU,ACCEL1,MUM,NUM,DM,DE,dt,STPALT,TT ,VTN,VR0)
      IMPLICIT REAL*16 (A-Z)
      CHARACTER*24 TITLED
      character*4 AD
      CHARACTER*1 BD, CD
      INTEGER VTN, DIRECT,NUM,VRN, AU,I
      DIMENSION X(4),VR(10),VT(5),RJAC(3)
      AU = 5
      WRITE (AU,71)
      WRITE (AU,72)
      AU = 1
      WRITE (AU,71)
      WRITE (AU,72)
   71 FORMAT ('1Trajectory Generation Model for Low-Thrust OTV''s',/,
     * ' in Cislunar Space using a Thrusting Control',/,
     *' Algorithm in the Restricted three-body')
   72 FORMAT (' formulation of the Earth-Moon system. ',/,
     *' Eagle Engineering, Inc.  (LSPI - djk)')
      AU = 5
      PRINT *, ' '
      PRINT *, 'OUTPUT WILL GO TO FILE "CISLUNAR.OUT"'
      WRITE(AU,7) SCMASS
    7 FORMAT('0Spacecraft initial mass    =',F10.2 )
      WRITE(AU,17) Isp
   17 FORMAT( ' Specific Impulse of engine = ', F8.2)
      WRITE(AU,27) Mdot
   27 FORMAT (' Mass flow rate of engines  = ',E15.8)
      WRITE(AU,37)
   37 FORMAT ('0Do you wish to specify s/c characteristics?  (y or n)')
      READ *, BD
      IF (BD.EQ.'y'.OR.BD.EQ.'Y') THEN
        PRINT *, ' Input Spacecraft initial mass.'
        READ  *, SCMASS
        PRINT *, ' Input Specific Impulse of engine.'
        READ  *, Isp
        PRINT *, ' Input Mass flow rate of engines.'
        READ  *, Mdot
        PRINT *, ' Input Degree of polynomial curve fit (2-7)'
        READ  *, VRN
      ENDIF
      THRUST = gravity * Isp * Mdot
      WRITE(AU,47)
   47 FORMAT ('0Starting Orbit About the Earth or Moon?' )
      READ  *, AD
      PRINT *, 'Input Destination Altitude at which to stop processing.'
```

```
      READ  *, STPALT
      PRINT *,' '
      PRINT *,' '
      PRINT *,'  PLEASE WAIT,  GENERATING PARAMETRIC VELOCITY EQN.'
      IF (AD.EQ.'m'.OR.AD.EQ.'M'.OR.AD.EQ.'moon')  AD='MOON'
      IF (AD.EQ.'MOON') THEN
       DIRECT = 1
       RANGE = 110000.
       TITLED='Moon to Earth Trajectory'
       CALL ERASE (VR, VT, VR0)
       X(1) = 0.0
       X(2) = 7178.
       X(3)=-QSQRT(MUE/X(2))
       X(4)= 0.0
       RANGESC = 225000.0
       CALL SPIRAL (X,DIRECT,RANGESC ,VT,VR,MUE,MUM,SCMASS,
     +     dt,THRUST,TT,Mdot,MU,ACCEL1,NUM,DM,DE,VRN,VTN,VR0)
       PRINT *, 'Input the radius from the Moon''s Center'
       READ  *, rl
       PRINT *, 'Input angle (deg) from -x axis'
       READ  *, theta
       PRINT *, 'Do you wish to specify the velocity? (y or n)'
       READ  *, BD
       IF (BD.EQ.'y'.OR.BD.EQ.'Y') THEN
        WRITE(AU,57)
 57     FORMAT ('0Input absolute velocity ' )
        READ *,VEXP
       ELSE
        vexp = -QSQRT(MUM/rl)
       ENDIF
       thet = theta * PI / 180.
       X(1) = rl * QCOS(thet) + DM
       X(2) = rl * QSIN(thet)
       X(3) = -vexp * QSIN(thet)
       X(4) = vexp * QCOS(thet)
       PRINT *, 'Modify Control Jacobians and Range?'
       READ  *, CD
       IF (CD.EQ.'Y'.OR.CD.EQ.'y') THEN
        WRITE(AU,67)
 67     FORMAT ('0Default Values 3.05, 2.82, 2.50')
        PRINT *,' JAC(1) = '
        READ  *,  RJAC(1)
        PRINT *,' Jac2 = '
        READ  *,  RJAC(2)
        PRINT *,' Jac3 = '
        READ  *,  RJAC(3)
        PRINT *,' Range = '
```

```
        READ *, RANGE
       ELSE
        RJAC(1)=3.05
        RJAC(2) = 2.82
        RJAC(3) = 2.5
       ENDIF
      ELSE
       DIRECT = 0
       RANGE = 255000.0
       TITLED = 'Earth to Moon Trajectory'
       CALL ERASE (VR,VT,VR0)
       X(1) = 0.0
       X(2) = 1838.
       X(3) = -QSQRT(MUM / X(2))
       X(4) = 0.
       RANGESC = 120000.0
       CALL SPIRAL (X, DIRECT, RANGESC, VT, VR, MUE, MUM,
     +      SCMASS, dt, THRUST, TT, Mdot,
     +      MU,ACCEL1,NUM, DM , DE,VRN ,VTN,VR0)
       PRINT *, 'Input the radius from the Earth''s center '
       READ *, rl
       PRINT *, 'Input angle (deg) from -x axis '
       READ *, theta
       PRINT *, 'Do you wish to specify the velocity? (y or n) '
       READ *, BD
       IF (BD.EQ.'Y'.OR.BD.EQ.'y') THEN
        WRITE(AU,77)
 77     FORMAT ('0Input absolute velocity ' )
        READ *, vexp
       ELSE
        vexp = QSQRT(MUE / rl)
       ENDIF
       thet = theta * PI / 180.
       X(1) = rl * QCOS(thet) -DE
       X(2) = rl * QSIN(thet)
       X(3) = -vexp * QSIN(thet)
       X(4) = vexp * QCOS(thet)
       PRINT *, 'Modify Control Jacobians and Range? '
       READ *, CD
       IF (CD.EQ.'Y'.OR.CD.EQ.'y') THEN
        WRITE(AU,87)
 87     FORMAT ('0Default Values 4.93, 4.10, 2.70' )
        PRINT *,' Jac1 = '
        READ *, RJAC(1)
        PRINT *, ' Jac2 = '
        READ *, RJAC(2)
        PRINT *, ' Jac3 = '
```

```
      READ  *, RJAC(3)
      PRINT *, ' Range = '
      READ  *, RANGE
    ELSE
      RJAC(1) = 4.93
      RJAC(2) = 4.1
      RJAC(3) = 2.7
    ENDIF
  ENDIF
  RETURN
  END
```

JACOBI Subroutine Description

The *JACOBI* subroutine calculates the instantaneous Jacobian constant of the spacecraft during its flight. This constant is then compared to a series of user defined control values to determine the necessary guidance control action.

JACOBI Internal and Shared Variables

CALL JACOBI (X(), CJ) STATIC

CJ	The instantaneous Jacobian constant of the spacecraft.
EN	The non-dimensionalized energy of the spacecraft in the three-body system.
RMT	Radial distance of the spacecraft from the center of the Moon (km).
ROEN	The non-dimensionalized distance the spacecraft is away from the Earth.
ROMN	The non-dimensionalized distance the spacecraft is away from the Moon.
RR	Radial distance of the spacecraft from the center of the Earth (km).
VELN	The non-dimensionalized velocity of the spacecraft.
VV	Magnitude of the spacecraft's velocity (km/s).
X()	State Vector of the spacecraft's position and velocity in the rotating x,y coordinates (km and km/s).
XN	The non-dimensionalized x-component of the spacecraft's position.
YN	The non-dimensionalized y-component of the spacecraft's position.

BASIC CODE

```
DEFDBL A-H, J-Z
SUB JACOBI (X(), CJ) STATIC
SHARED VV, RMT, RR
ROEN = RR / EMDIST
ROMN = RMT / EMDIST
XN = X(1) / EMDIST
YN = X(2) / EMDIST
EN = XN ^ 2 + YN ^ 2 + 2# * (1# - MUN) / ROEN + 2# * MUN / ROMN
VELN = VV / (NW * EMDIST)
CJ = EN - VELN * VELN + MUN * (1# - MUN)
END SUB
```

FORTRAN CODE

```
SUBROUTINE JACOBI (X,CJ, RR, EMDIST, RMT, MUN, VV, NW)
IMPLICIT REAL*16 (A-Z)
DIMENSION X(4), VR(10), RJAC(3) , VT(5)
ROEN = RR / EMDIST
ROMN = RMT / EMDIST
XN = X(1) / EMDIST
YN = X(2) / EMDIST
EN = XN ** 2 + YN ** 2 + 2. * (1. - MUN) /ROEN +2.*  MUN / ROMN
VELN = VV / (NW * EMDIST)
CJ = EN - VELN * VELN + MUN * (1. - MUN)
RETURN
END
```

POLYFIT SUBROUTINE DESCRIPTION

The *POLYFIT* subroutine is a stand along polynomial curve fitting routine that was taken from a Public Domain library of programs. It can fit up to seventh-order polynomial curves to the input data.

POLYFIT SUBROUTINE PASSED VARIABLES

CALL POLYFIT (PR(), PRV(), VRN, J, VR())

SUB POLYFIT (PR(), PV(), DEGREE, N, VR())

DEGREE	The degree of the polynomial curve for the data.
N	The number of data points.
PR()	The array of the sampled radial positions.
PV()	The array of the radial velocity component at each sampled position.
VR()	The output array of the coefficients for the polynomial curve.

BASIC CODE

```
DEFDBL A-H, J-Z
   SUB POLYFIT (PR(), PV(), DEGREE, N, VR())
   REDIM A(21), R(13, 14), t(14)
   D = DEGREE: A(1) = N
   FOR I = 1 TO N
    X = PR(I): Y = PV(I)
    FOR J = 2 TO 2 * D + 1
    A(J) = A(J) + X ^ (J - 1)
    NEXT J
    FOR K = 1 TO D + 1
    R(K, D + 2) = t(K) + Y * X ^ (K - 1)
    t(K) = t(K) + Y * X ^ (K - 1)
    NEXT K
    t(D + 2) = t(D + 2) + Y ^ 2
    NEXT I
    FOR J = 1 TO D + 1
    FOR K = 1 TO D + 1
    R(J, K) = A(J + K - 1)
    NEXT K
    NEXT J
    FOR J = 1 TO D + 1
    K = J
280 IF R(K, J) <> 0 THEN 320
    K = K + 1
    IF K <= D + 1 THEN 280
    PRINT "NO UNIQUE SOLUTION"
    GOTO 790
320 FOR I = 1 TO D + 2
    S = R(J, I)
    R(J, I) = R(K, I)
    R(K, I) = S
    NEXT I
    Z = 1 / R(J, J)
    FOR I = 1 TO D + 2
    R(J, I) = Z * R(J, I)
    NEXT I
    FOR K = 1 TO D + 1
    IF K = J THEN 470
    Z = -R(K, J)
    FOR I = 1 TO D + 2
    R(K, I) = R(K, I) + Z * R(J, I)
    NEXT I
470 NEXT K
    NEXT J
    PRINT
    PRINT "                CONSTANT ="; R(1, D + 2): CONSTA = R(1, D + 2): VR(0) = 
ONSTA
    FOR J = 1 TO D
    PRINT J; "DEGREE COEFFICIENT ="; R(J + 1, D + 2): VR(J) = R(J + 1, D + 2)
    NEXT J
    PRINT
    P = 0
    FOR J = 2 TO D + 1
    P = P + R(J, D + 2) * (t(J) - A(J) * t(1) / N)
    NEXT J
    q = t(D + 2) - t(1) ^ 2 / N
    Z = q - P
    I = N - D - 1
    PRINT
    J = P / q
    PRINT "COEFFICIENT OF DETERMINATION (R^2) = "; J
    PRINT "COEFFICIENT OF CORRLELATION ="; SQR(ABS(J))
    PRINT "STANDARD ERROR OF ESTIMATE ="; SQR(ABS(Z / I))
790 PRINT "POLYFIT COMPLETED"
  END SUB
```

FORTRAN CODE

```
      SUBROUTINE POLYFIT (PR, PV, DEGREE, N, VR,VR0)
      IMPLICIT REAL*16 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      INTEGER D , DEGREE, AU
      DIMENSION A(21),R(13,14),t(14),VR(10),PR(2100),PV(2100)
      D = DEGREE
      AU = 5
      A(1) = QFLOAT( N )
      DO 10 I = 1 , N
       X = PR(I)
       Y = PV(I)
       DO 20  J = 2, 2 * D + 1
        A(J) = A(J) + X ** (J - 1)
 20     CONTINUE
       DO 30 K =1,D+1
        R(K, D + 2) = t(K) + Y * X ** (K - 1)
        t(K) = t(K) + Y * X ** (K - 1)
 30     CONTINUE
       t(D +2)=t(D + 2) + Y ** 2.
 10    CONTINUE
      DO 40 J = 1, D + 1
       DO 50 K = 1, D + 1
        R(J, K) = A(J + K -1)
 50     CONTINUE
 40    CONTINUE
      DO 100 J = 1, D + 1
       K = J
280     IF (R(K, J).NE. 0.0) GOTO 320
       K = K + 1
       IF (K.LE. D + 1 ) GOTO 280
       WRITE(AU,7)
 7      FORMAT (' NO UNIQUE SOLUTION')
       GOTO 790
320     DO 60 I = 1, D + 2
        S = R(J, I)
        R(J, I) = R(K, I)
        R(K, I) = S
 60     CONTINUE
       Z = 1.0 / R(J, J)
       DO 70 I = 1 , D + 2
        R(J,I) = Z * R(J, I)
 70     CONTINUE
       DO 470 K = 1, D + 1
        IF (K.EQ.J) GOTO 470
        Z = -R(K,J)
        DO 90  I = 1, D + 2
        R(K, I) = R(K, I) + Z * R(J, I)
```

```
90    CONTINUE
470  CONTINUE
100 CONTINUE
    AU = 1
    WRITE(AU,17) R(1, D+2)
17  FORMAT ('0  CONSTANT         =',E15.8)
    CONSTA = R(1, D + 2)
    VR0 = CONSTA
    DO 110 J = 1 , D
     WRITE(AU,27) J, R(J+1,D+2)
27   FORMAT(1X,I4, 'DEGREE COEFFICIENT =',E15.8)
     VR(J) = R(J + 1, D + 2)
110 CONTINUE
    P = 0
    DO 120 J = 2 ,D + 1
     P = P + R(J, D + 2) * (t(J) - A(J) * t(1) / QFLOAT( N ) )
120 CONTINUE
     q = t(D + 2) - t(1) ** 2. / QFLOAT( N )
     Z = q - P
     I = N - D - 1
     PJ= P / q
     WRITE(AU,37) PJ
37   FORMAT (' COEFFICIENT OF DETERMINATION (R^2)=',E15.8 )
     SQAJ = QSQRT( QABS( PJ ) )
     SQAZI = QSQRT(QABS(Z/QFLOAT(I)))
     WRITE(AU,47) SQAJ
47   FORMAT (' COEFFICIENT OF CORRELATION       =',E15.8)
     WRITE(AU,57) SQAZI
57   FORMAT (' STANDARD ERROR OF ESTIMATE       =',E15.8)
790  PRINT *,'POLYFIT COMPLETED'
     RETURN
     END
```

RUK and RUK4 Subroutine Descriptions

The *RUK* and *RUK4* subroutine are fourth order Runge-Kutta integration routines. They call *D ER1* and *DER*, respectively, and integrate the equations of motion of the spacecraft.

RUK and RUK4 Subroutine Passed Variables

SUB RUK (X(), N, dt) STATIC

SUB RUK4 (X(), N, dt) STATIC

dt Integration step size (seconds).

NUM Order of the X state vector for the Runge-Kutta.

X() State Vector of the spacecraft's position and velocity (km and km/s).

BASIC CODE

```
DEFDBL A-H, J-M, O-Z
SUB RUK4 (X(), N, dt) STATIC
DIM D(6), F(6), U(6), DX(6)
CALL DER(X(), D())
FOR I = 1 TO N
        D(I) = D(I) * dt: U(I) = X(I) + .5# * D(I)
NEXT I
CALL DER(U(), F())
FOR I = 1 TO N
        F(I) = F(I) * dt: D(I) = D(I) + 2# * F(I): U(I) = X(I) + .5# * F(I)
NEXT I
CALL DER(U(), F())
FOR I = 1 TO N
        F(I) = F(I) * dt: D(I) = D(I) + 2# * F(I): U(I) = X(I) + F(I)
NEXT I
CALL DER(U(), F())
FOR I = 1 TO N
        X(I) = X(I) + (D(I) + F(I) * dt) / 6#
NEXT I
END SUB
```

FORTRAN CODE

```
      SUBROUTINE RUK4 (X, N, dt, MU, ACCEL1,DM,DE,MUM,
     *        MUE,TH,VR,thrst,CJ,ACCEL,RANGE,DIRECT,ALPHA,
     *  RANGEOFF,theta,RJAC,VTN,DIST,VRR,VRN,VR0,TT,VT)
      IMPLICIT REAL*16 (A-Z)
      INTEGER I,N,TH,thrst,VTN,VRN
      DIMENSION D(4),F(4),U(4),X(4),VR(10),VT(5),RJAC(3)
      CALL DER(X, D,DM,DE,MUM,MUE,TH,VR,thrst,CJ,ACCEL,RANGE,
     *  DIRECT,ALPHA,RANGEOFF,theta,RJAC,VTN,VT,DIST,VRR,VRN,VR0,TT)
      DO 10 I = 1 , N
      D(I) = D(I) * dt
      U(I) = X(I) + .5 * D( I )
 10   CONTINUE
      CALL DER(U, F,DM,DE,MUM,MUE,TH,VR,thrst,CJ,ACCEL,RANGE,
     *  DIRECT,ALPHA,RANGEOFF,theta,RJAC,VTN,VT,DIST,VRR,VRN,VR0,TT)
      DO 20 I = 1, N
      F(I) = F(I) * dt
      D(I) = D(I) + 2. * F(I)
      U(I) = X(I) + .5 * F(I)
 20   CONTINUE
      CALL DER(U, F,DM,DE,MUM,MUE,TH,VR,thrst,CJ,ACCEL,RANGE,
     *  DIRECT,ALPHA,RANGEOFF,theta,RJAC,VTN,VT,DIST,VRR,VRN,VR0,TT)
      DO 30 I = 1 , N
      F(I) = F(I) * dt
      D(I) = D(I) + 2. * F(I)
      U(I) = X(I) + F(I)
 30   CONTINUE
      CALL DER(U, F,DM,DE,MUM,MUE,TH,VR,thrst,CJ,ACCEL,RANGE,
     *  DIRECT,ALPHA,RANGEOFF,theta,RJAC,VTN,VT,DIST,VRR,VRN,VR0,TT)
      DO 40 I = 1,N
        X(I) =X(I) + (D(I) + F(I) *dt)/6.
 40   CONTINUE
      RETURN
      END
```

SPIRAL Subroutine Description

The subroutine *SPIRAL* is used to generate the parametric curves of the radial and tangential velocity components for the reference capture spiral. The direction of the cislunar trajectory determines the governing gravitational parameter, MU, for the spiral trajectory generation. The estimated final mass about the target planet is taken to be 80% of the spacecraft's chosen initial mass. The subroutine begins an integration loop using a negative mass flow for the spacecraft's propulsion system. A spiral trajectory is generated out from the target planet of the spacecraft gaining mass as it goes. This mimics the ideal spiral capture with the spacecraft losing mass as it settles into the capture orbit. The integration routine is a fourth order Runge-Kutta that calls the derivative subroutine, *DER1*. *DER1* contains the equations of motion for a two-body trajectory. No perturbational forces, other than the spacecraft's acceleration, are included in the two-dimensional equations of the motion. *SPIRAL* calculates the acceleration level, the two-body energy, and the tangential and radial velocity components. These velocity components are determined by finding the angle, theta, between the radial vector and the velocity vector using the dot products rule,

```
r x v = |r||v| cos(theta)
```

where r is the radius, and v is the velocity. The radial component of the velocity is found by multiplying the cosine of theta by *VMAG*, the magnitude of the velocity vector. The tangential component of the velocity vector is similarly found by multiplying *VMAG* by the sin of theta. Every tenth integration the velocity components and radial distance is captured into three arrays, PVR(), PVT(), and PR(). When the two-body energy of the spiral trajectory is non-negative the spacecraft's thrust is turned off and the spacecraft is assumed to be following a parabolic path. The integration continues until the spacecraft passes the range flag, *RANGESC*. *POLYFIT* develops a polynomial curve fit of the radial velocity component. The tangential velocity component is fit to either a power, exponential, logarithmic, or linear curve in the subroutine *CURVE* depending on which equation best fits the data. When the parametric curve fitting is complete the program records the velocity curve coefficients into VR() and VT() and returns control to IO.

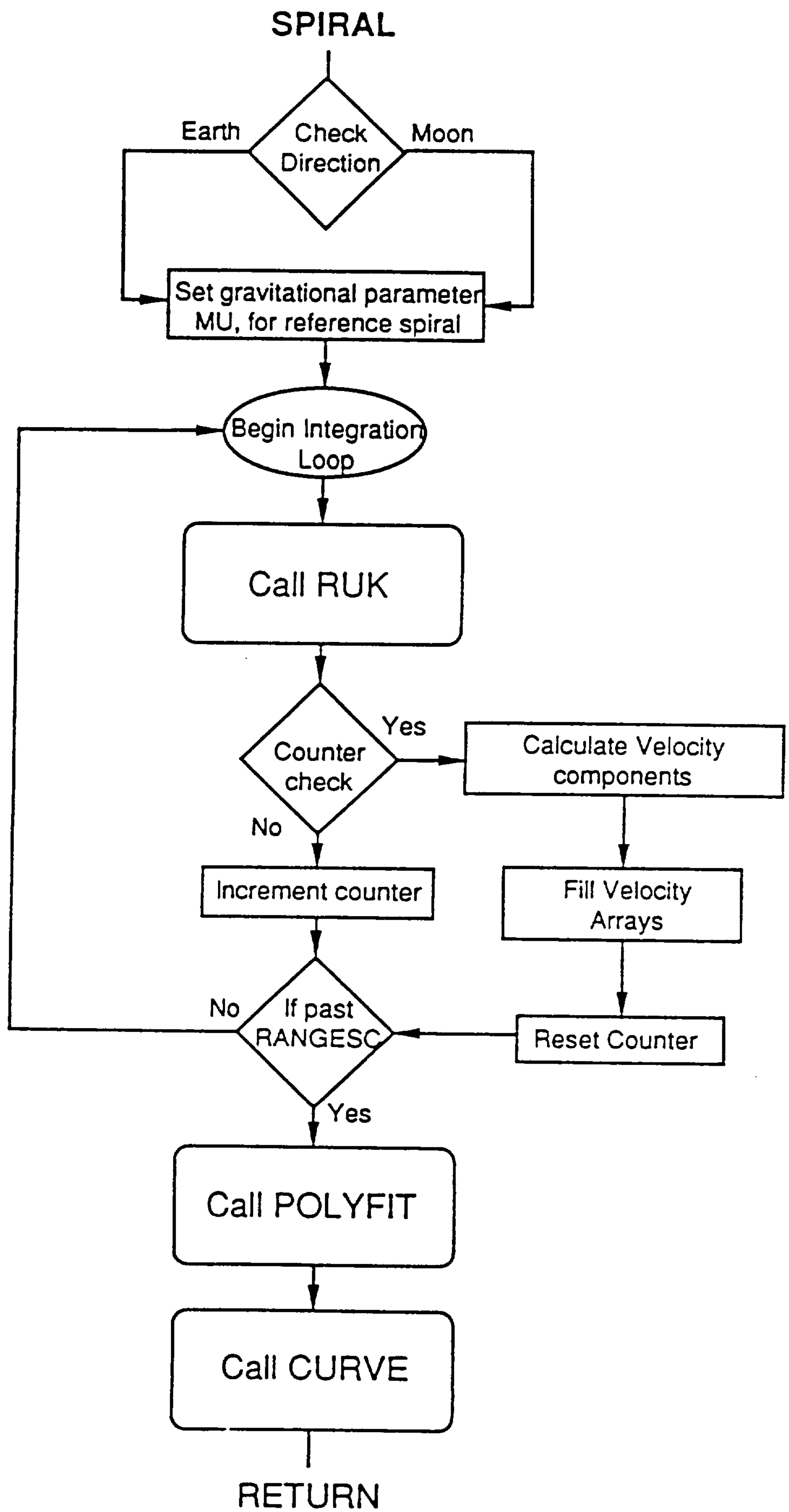
SPIRAL Subroutine Internal and Shared Variables

SUB SPIRAL (X(), DIRECTION, RANGESC, VT(), VR()

ctheta	Cosine of the angle between the radius and velocity vector.
DIRECTION	Numeric indicator of the direction of the trajectory generation, Earth to Moon (0) or Moon to Earth (1).
dt	Integration step size (seconds).
e	Two-body energy, sum of the potential and kinetic energy.
J	The array size counter for the radial and tangential velocity, and the radial distance arrays.
PR()	The array containing the radial distance from the capture planet (km/1000).
PVR()	The array containing the radial component of velocity, vrad, during the reverse integration spiral (km/s).
PVT()	The array containing the tangential component of velocity, vtan, during the reverse integration spiral (km/s).
RANGESC	Range from the central body for the reverse integration spiral to be generated before stopping (km).
RDOTV	Radial vector dot multiplied with the velocity vector.
RMAG	Magnitude of the spacecraft's radius vector from the capture planet (km).
SCMASS1	Starting mass for reverse integration of spacecraft from target planet, 80% of chosen initial mass.
TT	Total time of trajectory generation (seconds).
VMAG	Magnitude of the spacecraft's velocity vector (km/s).
VMAG2	Square of the velocity magnitude of the spacecraft in relation to the capture planet (km/s)^2.
VR()	The array of the parameterized radial velocity coefficients.
vrad	Component of the spacecraft's velocity in the radial direction (km/s).
VT()	The array of parameterized tangential velocity coefficients.

vtan	Component of the spacecraft's velocity in the tangential direction (km/s).
vthet	Angle between the radius and velocity vector (radians).
w	Counter for sampling the capture spiral trajectory.
X()	State Vector of the spacecraft's position and velocity in the rotating x,y coordinates (km and km/s).

SPIRAL Subroutine Flowchart



BASIC CODE

```
DEFDBL A-Z
SUB SPIRAL (X(), DIRECTION, RANGESC, VT(), VR())
REDIM PR(2100), PVR(2100), PVT(2100)
IF DIRECTION = 1 THEN
        MU = MUE
        ELSE
        MU = MUM
END IF
I = 0: w = 5: J = 0: dt = 3
SCMASS1 = SCMASS * .8#                   'total s/c mass (kg)
SCREEN 2: WINDOW (70000, 50000)-(-70000, -50000)
CIRCLE (0, 0), X(2), 1
DO
   RUK X(), NUM, dt
   RMAG = SQR(X(1) ^ 2 + X(2) ^ 2)
   dt = .02# * RMAG
   TT = TT + dt: I = I + 1
   VMAG2 = X(3) ^ 2 + X(4) ^ 2
   e = VMAG2 / 2# - MU / RMAG
   ACCEL1 = THRUST / (SCMASS1 + Mdot * TT) 'acceleration of the s/c (km/s^2)
   IF (e >= 0#) THEN ACCEL1 = 0#
   IF w = 10 THEN
        VMAG = SQR(VMAG2)
        RDOTV = X(1) * X(3) + X(2) * X(4)
        ctheta = RDOTV / (RMAG * VMAG)
        vthet = ACOS(ctheta)
        vrad = VMAG * COS(vthet)
        vtan = VMAG * SIN(vthet)
        J = J + 1: SHOW X()
        PR(J) = RMAG / 1000#: PVR(J) = vrad: PVT(J) = vtan: w = 0
   END IF
   w = w + 1
LOOP WHILE RMAG < RANGESC
PRINT " done", RMAG, vrad, vtan, I, J, e
POLYFIT PR(), PVR(), VRN, J, VR()
PRINT "The coefficients of vrad vs rad polynomial"
CURVE PR(), PVT(), J, VT()
END SUB
```

FORTRAN CODE

```
      SUBROUTINE SPIRAL (X, DIRECT, RANGESC, VT, VR, MUE,
     +  MUM, SCMASS, dt, THRUST, TT, Mdot,
     +  MU, ACCEL1,NUM,DM,DE,VRN,VTN,VR0)
      IMPLICIT REAL*16 (A-H,O-Z)
      REAL*16 MU,MUM,MUE,Mdot
      INTEGER W,VRN,DIRECT,AU,VTN
      DIMENSION X(4), PR(2100), PVR(2100), PVT(2100), VR(10),VT(5)
      DIMENSION ABCD(100)
      AU = 5
      IF (DIRECT.EQ.1) THEN
        MU = MUE
      ELSE
        MU = MUM
      ENDIF
      w = 5
      J = 0
      dt = 3.0
      SCMASS1 = SCMASS * .8
      DO WHILE ( RMAG .LT. RANGESC )
        CALL RUK (X,NUM, dt,MU,ACCEL1)
        RMAG = QSQRT(X(1) ** 2. + X(2) ** 2.)
        dt = .02 * RMAG
        TT = TT + dt
        VMAG2 = X(3) ** 2. + X(4) ** 2.
        e = VMAG2 / 2. - MU / RMAG
        ACCEL1 = THRUST/(SCMASS1 + Mdot * TT)
        IF (e.GE.0.0) ACCEL1 =0.0
        IF (w.EQ.10) THEN
          VMAG = QSQRT(VMAG2)
          RDOTV = X(1) * X(3) + X(2) * X(4)
          ctheta = RDOTV / (RMAG * VMAG)
          vthet = QACOS(ctheta)
          vrad = VMAG * QCOS(vthet)
          vtan = VMAG * QSIN(vthet)
          J= J + 1
          PR(J)=RMAG/1000.0
          PVR(J) = vrad
          PVT(J) = vtan
          w=0
        ENDIF
        w= w+ 1
      END DO
      CALL POLYFIT (PR,PVR,VRN,J,VR,VR0)
      AU = 1
      WRITE(AU,107)
  107 FORMAT ('1The coefficients of vrad vs rad polynomial' )
      CALL CURVE(PR,PVT,J,VT,VR0,VTN)
```

```
      RETURN
      END
```